9-5-2013

# Survivable Cloud Networking Services

Feng Gu

Feng Gu
*Candidate*

Department of Electrical and Computer Engineering
*Department*

This dissertation is approved, and it is acceptable in quality and form for publication:

*Approved by the Dissertation Committee:*

Dr. Nasir Ghani , Chairperson

Dr. Wei Shu

Dr. Marios Pattichis

Dr. Khaled Bashir Shaban

# SURVIVABLE CLOUD NETWORKING SERVICES

by

**Feng Gu**

B.E., Huazhong University of Science and Technology, 2008
M.S., Computer Engineering, University of New Mexico, 2010

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2013

# Dedication

*To my parents*

# Acknowledgments

I would like to acknowledge and dedicate my gratitude to the following people who made the completion of this work possible:

My advisor, Dr. Nasir Ghani, for his continuous help and advisement through my PhD study.

My dissertation committee members, Dr. Wei Shu, Dr. Marios Pattichis, and Dr. Khaled Bashir Shaban, for their valuable suggestions and advice on this dissertation work.

My research colleagues, Dr. Qing Liu, Dr. Chongyang Xie, Dr. Feng Xu, Ms. Kaile Liang, and Mr. Hao Bai, for their help and collaboration in my research.

All my friends, for their great support and encouragement.

And most especially my parents, for their endless love and support.

# SURVIVABLE CLOUD NETWORKING SERVICES

by

**Feng Gu**

B.E., Huazhong University of Science and Technology, 2008
M.S., Computer Engineering, University of New Mexico, 2010

Doctor of Philosophy, Engineering, University of New Mexico, 2013

## Abstract

Cloud computing paradigms are seeing very strong traction today and are being propelled by advances in multi-core processor, storage, and high-bandwidth networking technologies. Now as this growth unfolds, there is a growing need to distribute cloud services over multiple data-center sites in order to improve speed, responsiveness, as well as reliability. Overall, this trend is pushing the need for *virtual network* (VN) embedding support at the underlying network layer. Moreover, as more and more "mission-critical" end-user applications move to the cloud, associated VN survivability concerns are also becoming a key requirement in order to guarantee user service level agreements.

Overall, several different types of survivable VN embedding schemes have been developed in recent years. Broadly, these schemes offer resiliency guarantees by pre-provisioning backup resources at service setup time. However, most of these solutions are only geared towards handling isolated single link or single node failures. As such, these designs are largely ineffective against larger regional stressors that can result

in multiple system failures. In particular, many cloud service providers are very concerned about catastrophic disaster events such as earthquakes, floods, hurricanes, cascading power outages, and even malicious weapons of mass destruction attacks. Hence there is a pressing need to develop more robust cloud recovery schemes for disaster recovery that leverage underlying distributed networking capabilities.

In light of the above, this dissertation proposes a range of solutions to address cloud networking services recovery under multi-failure stressors. First, a novel failure region-disjoint VN protection scheme is proposed to achieve improved efficiency for pre-provisioned protection. Next, enhanced VN mapping schemes are studied with probabilistic considerations to minimize risk for VN requests under stochastic failure scenarios. Finally, novel post-fault VN restoration schemes are also developed to provide viable last-gap recovery mechanisms using partial and full VN remapping strategies. The performance of these various solutions is evaluated using discrete event simulation and is also compared to existing strategies.

# Contents

# List of Figures

*List of Figures*

# Glossary

| | |
|---|---|
| *ASAP* | Application Specific and Agile Private |
| *BVNE* | Baseline VN Embedding |
| *C-NSVIM* | Connectivity-Aware NSVIM |
| *CPP* | Cluster and Path Protection |
| *DFRDM* | Dynamic Failure Region Disjoint Mapping |
| *D-ViNE* | Deterministic Rounding based Virtual Network Embedding |
| *E2C* | Elastic Compute Cloud |
| *EVN* | Enhanced VN |
| *FD-EVN* | Failure-Dependent EVN |
| *FMR* | Full Migration Restoration |
| *FRGBM* | Failure Region Group Based Mapping |
| *IaaS* | Infrastructure as a Service |
| *IT* | Information Technology |
| *ILP* | Integer Linear Programming |

*Glossary*

| | |
|---|---|
| *IOCM* | Incremental Optimization with Constrained Mapping |
| *J-PMR* | Joint-Partial Migration Restoration |
| *LP* | Linear Programming |
| *LB* | Load Balancing |
| *MC* | Mapping Cost |
| *MCF* | Multi-Commodity Flow |
| *MILP* | Mixed Integer Linear Programming |
| *MIP* | Mixed Integer Programming |
| *MP-SVIMA* | Migratory Protection-based Virtual Infrastructure Mapping Algorithm |
| *NSVIM* | Non-Survivable Virtual Infrastructure Mapping |
| *P2P* | Point-to-Point |
| *PaaS* | Platform as a Service |
| *PMR* | Partial Migration Restoration |
| *P-SVIMA* | Protection-Based Survivable Virtual Infrastructure Mapping Algorithm |
| *QoS* | Quality of Service |
| *SaaS* | Software as a Service |
| *SOD_BK* | Shared On-Demand |
| *SOUM* | Separate Optimization with Unconstrained Mapping |

| | |
|---|---|
| *SPA_BK* | Shared Pre-Allocation |
| *SRLG* | Shared Risk Link Group |
| *R-ViNE* | Randomized rounding based Virtual Network Embedding |
| *SLAs* | Service Level Agreements |
| *TE* | Traffic Engineering |
| *VM* | Virtual Machine |
| *VN* | Virtual Network |
| *VNE* | Virtual Network Embedding |
| *VNP* | Virtual Network Protection |
| *VPN* | Virtual Private Network |
| *WMD* | Weapons of Mass Destruction |

# Chapter 1

# Introduction

This dissertation addresses the topic of survivability in cloud networking systems. To properly introduce the work, this chapter first presents a brief overview of this space and then highlights the key motivations for the research. The main contributions are then presented along with an overview of the thesis chapters.

## 1.1 Background Overview

Cloud computing paradigms have gained rapid traction in recent years, driven by advances in multi-core processor, storage, and high-bandwidth networking technologies. These provisions allow organizations to outsource their *information technology* (IT) infrastructure and even software service needs to external cloud provider organizations. In turn these providers maintain extensive networked data-center facilities and leverage advanced network and machine virtualization techniques to provide a full range of customized service offerings. In particular, virtualization provides the necessary abstraction such that dispersed underlying physical resources (raw computation, storage, network bandwidth) can be unified into a common pool for a client.

Futhermore, virtualization also allows operators to deploy application services over these virtualized resource pools and simplify manageability/security concerns. In particular, *virtual machine* (VM) technology is now widely adopted in cloud computing environment as users can specify customized software suites e.g., operating systems, software applications, and pack them together into VMs.

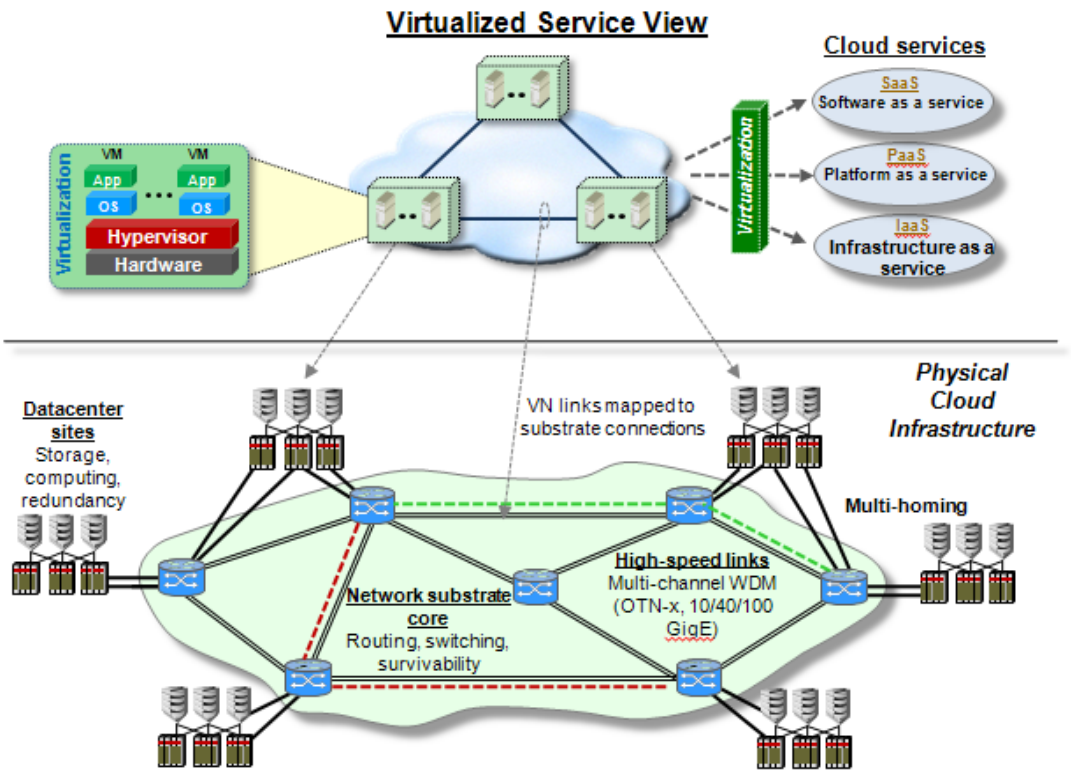



Figure 1.1: Cloud service model

Overall, cloud-based services offer many saliencies. Foremost, they allow users to forgo expensive data-center build-outs, yielding much lower capital and operational expenditures. In addition, they provide users with dynamic access to vast amounts of computational resources in a flexible pay-as-you-grow manner, i.e., service scalability. Finally, cloud-based offerings allow organizations to distribute their content and

operations across multiple dispersed sites via network/machine virtualization techniques, thereby improving survivability and responsiveness, i.e., geo-diversity [BR01], [AG01]. Leveraging these new capabilities many types of cloud service models have emerged in recent years, including *infrastructure as a service* (IaaS), *platform as a service* (PaaS), *software as a service* (SaaS) (see also Figure 1.1) [BR01]. Namely IaaS represents the lowest layer of service abstraction and directly provisions hardware resources for clients to support their applications. Meanwhile, PaaS offers a higher level of abstraction by providing an integrated platform for clients develop their customized applications on. Finally, SaaS offers full turnkey software applications that user can access remotely via usage-based pricing models [IF01]. Figure 1.2 shows some popular commercial cloud service offerings in use today.

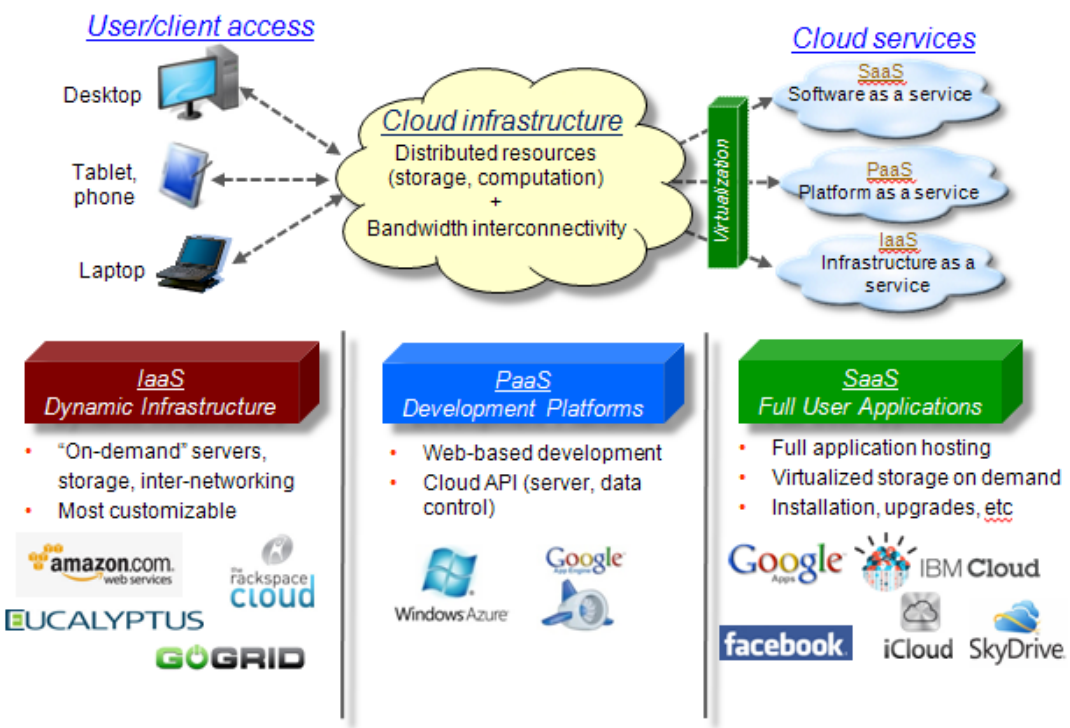



Figure 1.2: Overview of cloud network architecture

Now as new cloud offerings gain traction, there is a growing need to distribute their operation across multiple data-center sites in order to improve speed and responsiveness, as well as reliability [BR01]. As a result, underling networking substrates will play a critical role in provisioning high-speed interconnectivity between such data-center locations. In particular, the concept of *network virtualization* [YZ01] is very germane here as it allows operators to provision dedicated *virtual networks* (VN) overlays and interconnect cloud users (or providers) over common physical infrastructures [GS01]. In general, VN connectivity can be specified and varied according to particular client needs. An example of a cloud-based VN overlay is shown in Figure 1.1 for a sample 3 node ring topology embedded over a 7 node network interconnecting 6 data-center sites.

Overall, the topic of VN request provisioning over physical networking infrastructures has been well-studied in recent years [AF01], i.e., also termed as the *VN embedding* (VNE) problem. The main goal here is to embed VN requests onto physical network and data-center infrastructures to meet client needs. Namely, each VN node requires a certain amount of data-center computing and storage resources, whereas each VN link/edge requires a certain amount of bandwidth capacity to support communications between VN nodes. Hence operators must carefully provision their underlying node and link resources to map VN nodes to physical substrate data-center sites and VN links to underlying connections between sites. From an operational viewpoint, the key objectives here include revenue maximization and/or cost reduction and a full range of schemes have been developed using optimization and graph-based heuristic methods, see surveys in [AF01], [AB01].

Carefully note that studies have also looked at *virtual private network* (VPN) provisioning in IP (Layer 3), Ethernet (Layer 2), and even optical (Layer 1) networks, see [ZZ02]. Although somewhat related to the VNE problem for cloud services, VPN provisioning is slightly less complicated since VPN sites, i.e., connection end-points,

are fixed and pre-specified by clients. This obviates the need for VN node placement, a key dimensionality reduction.

## 1.2 Motivations

As cloud services expand and start to support more mission-critical end-user applications, underlying VN reliability is becoming a major concern. This is particularly important if users have established *service level agreements* (SLAs) that providers must meet. However, many cloud recovery schemes are based upon localized intra-site server and storage redundancies to support data and virtual server image backup [CZ01]. Although these schemes give very fast recovery, they are ineffective against large regional failure events, i.e., such as those arising from natural disasters, massive power outages, or malicious *weapons of mass destruction* (WMD) attacks. Namely these stressors can yield multiple system failures with high levels of spatial (geographic) and time correlation, quickly overwhelming localized recovery provisions and even impacting whole data-center sites. Hence there is a pressing need to develop more robust cloud recovery schemes for disaster conditions that leverage underlying distributed networking capabilities. This issue has gained added impetus owing to some recent high-profile cloud services outages, e.g., the Amazon *Elastic Compute Cloud* (E2C) outage in April 2011 affected many user services for several days [WP01].

Now from the perspective of network survivability, the topic of point-to-point connection recovery between fixed end points has been very well-studied, see [SS01]. In particular, most related solutions have proposed pre-provisioned strategies to reserve dedicated/shared backup resources ahead of time, i.e., protection. However, the broader VN survivability problem is much more involved owing to the higher dimensionalities of the VNE problem. Namely, since a VN instance is comprised of

multiple nodes and connections mappings, related recovery schemes have to consider both connection recovery and (failed) connection end-point re-mapping.

Overall, VN survivability is a relatively new area, and one that has seen limited attention to date. Akin to connection recovery schemes, most VN survivability solutions have also outlined pre-provisioned protection strategies. For example, some researchers have looked at single link failure recovery in [HY01], [XZ01], [YC01], [MR01] and [TG01]. Similarly, [HY02], [HY03] and [CQ01] have also studied improved VN mappings to handle single node failure recovery. However, these solutions are largely ineffective against regional failures causing widespread correlated node/link outages. Indeed, the only known work on VN recovery under more challenging regional failures is presented in [HY04] and [GS02]. Again, these algorithms propose guaranteed protection schemes, but pre-compute separate backup VNs for each potential failure event. Clearly, this leads to excessive resource consumption, and hence the authors have also proposed resource sharing between backup VN mappings. Although these efforts represent some important initial contributions, they are generally resource-intensive and yield higher costs. Moreover, they do not account for the probabilistic nature of regional disasters/stressors. In such cases, full recovery guarantees are not very feasible due to the excessive number of potential node/link failure combinations that have to be accounted for.

In light of the above, this thesis work is motivated by the growing need to develop improved network survivability solutions to support cloud services against large-scale catastrophic failures. Indeed such disaster scenarios are a key concern (inhibitor) for many organizations looking to migrate their core operations into the cloud. Carefully note that this thesis will only focus on inter-site (data-center) networking recovery solutions. Although, many intra-site recovery schemes are also being used with data-centers themselves, these solutions are designed for specialized internal data-center topologies and again, offer little resiliency against large-scale disaster type events,

see [MA01] for details.

## 1.3 Problem Statement

This dissertation is motivated by the growing need to develop robust disaster recovery solutions for cloud-computing paradigms. Specifically, new VN survivability schemes are required to provide improved protection recovery without imposing excessive resource usage overheads. In addition, new solutions are also needed to handle the probabilistic nature of disaster events. Finally, post-fault restoration schemes also need to be investigated as they provide a very viable last-gap alternative in case of pre-provisioned protection failure.

To address these concerns, this dissertation develops novel solutions for survivable VN mapping under large regional failure scenarios. The overall focus is to develop efficient VN survivable mapping schemes which can handle multiple cloud systems failures with improved resource efficiency and reliability. The performance of these proposed schemes is further evaluated and analyzed using both optimization and network simulation techniques.

## 1.4 Proposed Work

This dissertation addresses a range of open issues in the area of cloud networking survivability. In particular, the key contributions here include the following:

1) Novel pre-provision VN protection mapping schemes to guarantee multi-failure recovery against disjoint regional node/link failures, 2) novel probabilistic VN risk minimization mapping schemes to minimize failure risk while also taking into account resource efficiency concerns, 3) efficient post-fault VN restoration schemes to re-

compute affected VN mapping after multi-failure stresses

Overall, the rest of this dissertation is organized as follows. Chapter 2 presents a detailed survey of the latest key work in VNE design and survivability. Next, Chapter 3 details a novel VN protection scheme to recover from multi-failure stressor events. Extending upon this, Chapter 4 treats disasters as stochastic (random) events and presents a novel solution to lower VN failure risks. Chapter 5 then looks at the design of post-fault restoration schemes to restore failed VN mappings. Finally, detailed conclusions and directions for future work are presented in Chapter 6.

# Chapter 2

# Background and Related Work

The provisioning of cloud networking services is an area of growing interest today. Along these lines, this chapter presents a review of some of the latest work in this field, focusing on regular VN embedding as well as survivable VN embedding schemes. Open research challenges are then outlined to motivate the thesis research.

## 2.1 VN Embedding Design (Non-Survivable)

Various studies have looked at mapping user VN requests over physical substrate networks, i.e., VNE problem. Here, both the VN requests and the substrate networks are denoted by graphs, with each VN node usually requiring a certain amount of nodal resources and each VN edge requiring a certain amount of bandwidth capacity. An example of VN embedding is shown in Figure 2.1, where two 3-node VN requests are mapped onto a 10-node substrate network (and the respective numbers next to the nodes/links represent their associated capacities). Here the sample VN link from node $a$ to $c$ is routed as a network connection between substrate nodes 1-3-6-4-5. Now earlier work in [NC01] has shown the VNE problem to be NP-hard. Hence, most

studies have proposed optimization and heuristics-based strategies to try to minimize substrate network resource usages or maximize carrier revenues, see [YZ01], [MY01], [NC01], [JL01], [XC01], [HY05]. Some of these schemes are now reviewed and a high level summary is also presented in Figure 2.2.

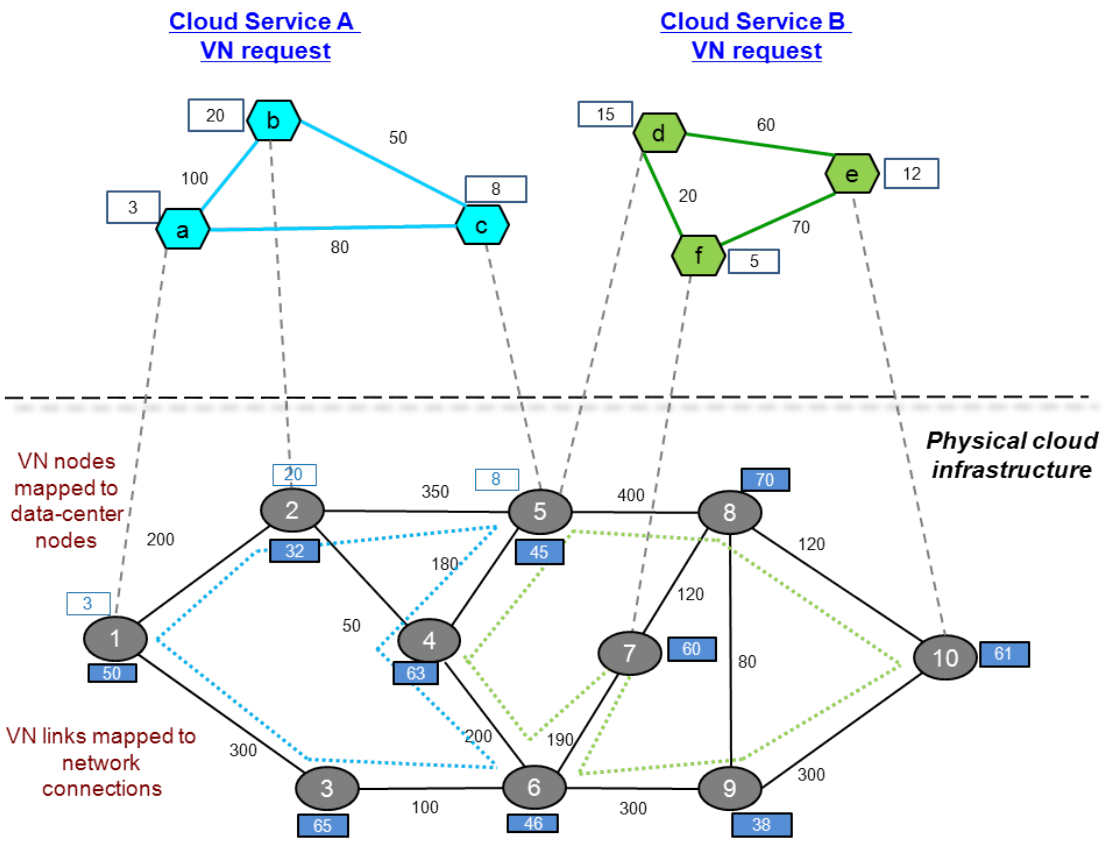



Figure 2.1: A VN embedding example

### 2.1.1 Optimization-Based Solutions

*Linear programming* (LP) [RA01] techniques have been widely used to find optimal solutions for a broad range of networking problems. However, the inherent characteristics of the VNE problem mandate integral constraints for many of the variables

in an LP formulation. As a result, the VN mapping problem is usually formulated as a more specialized *integer linear programming* (ILP) [ZZ01] problem which only permits integer variables and precludes path splitting for VN link connections. Alternatively, some studies have also proposed *mixed integer linear programming* (MILP) [NC01] formulations to relax the path-splitting constraint, i.e., both integer and real variables. Consider some further details here.

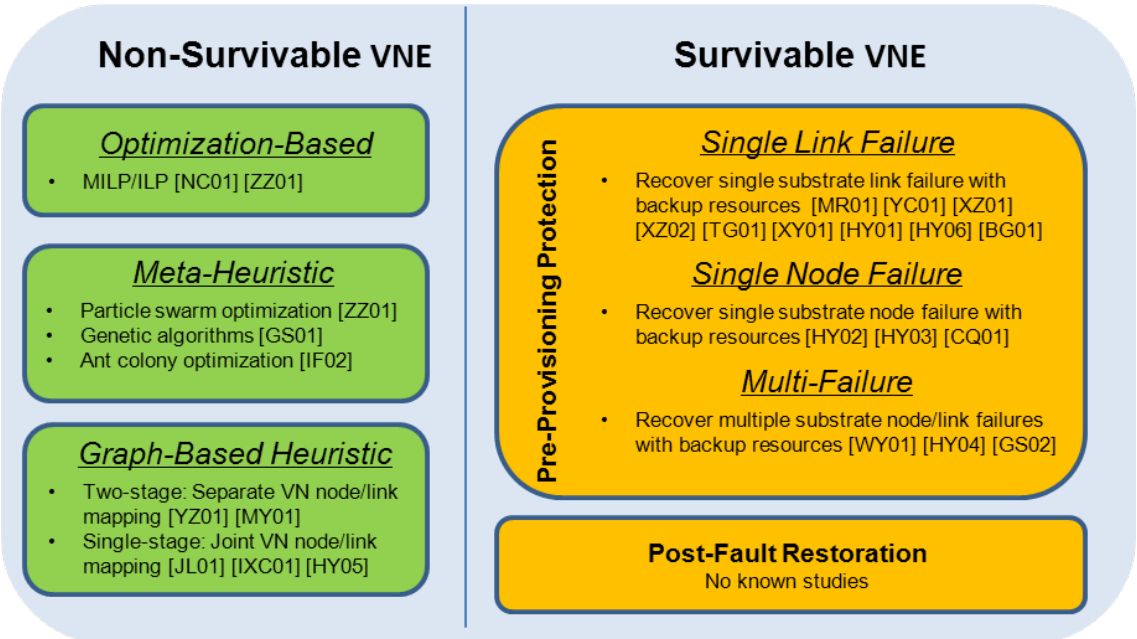



Figure 2.2: A summary of VNE provisioning and survivability schemes

A well-known MILP approach for VNE is presented in [NC01]. In order to handle the VN node mapping problem, here each VN node is treated as a "meta-node" and connected to all substrate nodes via special *virtual* infinite-capacity "meta-links". Additional constraints are also added to restrict each meta-node to only use one of its adjacent meta-links, i.e., since a VN node can only be mapped to one unique substrate node. Hence these auxiliary meta-nodes and links essentially connect the VN topology with the substrate network and allow the VNE problem to be formulated over a single network (instead of two separate ones). However, the resultant

MILP formulation is quite complex and difficult to solve. As a result, the authors also propose several relaxation techniques [NC01]. In particular, the *deterministic rounding based virtual network embedding* (D-ViNE) algorithm relaxes the integral constraints for the meta-node mapping variables, thereby reducing the formulation to a *multi-commodity flow* (MCF) problem [RA01], i.e., solvable in polynomial time. By computing the product of each relaxed meta-node mapping variable and the amount of flows on that meta-link, the VN node mapping can then determined as the one with the maximum product value. Finally, once the VN node mappings have been computed, VN link mappings are derived by solving another MCF problem. A slightly-modified *randomized rounding based virtual network embedding* (R-ViNE) relaxation scheme is also proposed in [NC01]. The only difference here is that instead of selecting the maximum product value (as per D-ViNE), this scheme normalizes each product value to the [0,1] range before selecting the mapping solution. Both relaxation schemes are evaluated for a sample 50-node network, and overall results shows that R-ViNE slightly outperforms D-ViNE in terms of acceptance ratios and average revenues, but yields slightly higher average costs.

An alternate ILP-based VNE solution is also proposed in [ZZ01]. This scheme does not utilize the concept of meta-nodes and meta-links, and instead uses a binary variable to determine the mapping between a VN node and a substrate node. Therefore, this (ILP) formulation greatly reduces the number of variables needed as compared to the MILP solution in [NC01]. However, the ILP formulation is not solved, and instead the authors propose a meta-heuristic solution (detailed next in Section 2.1.2).

### 2.1.2 Meta-Heuristic Solutions

In general, finding optimal solutions for large combinatorial problems is quite difficult (computationally intractable). As a result, researchers have also applied various meta-heuristic schemes to find "near-optimal" solutions by attempting to improve a candidate solution toward the optimal solution. Now within the context of VNE design, some key strategies include simulated annealing [SK01], genetic algorithms [JH01], ant colony optimization [TS01], particle swarm optimization [JK01], and tabu search techniques [FG01]. Some of these solutions are now surveyed.

The work in [ZZ01] solves the VNE problem by using particle swarm optimization. Namely, a potential VN node mapping solution is represented using a position vector of a particle, and a velocity vector for this particle is also defined to adjust the VN node mapping. A number of such particles (potential mappings) are then initialized by selecting starting positions, and these positions are then adjusted in an iterative manner. A fitness value is also introduced to gauge each position, i.e., VN mapping choice. Namely, VN link mappings are computed for each position using shortest-path algorithms. Here, if a VN link mapping fails, then the position of the particle is re-initialized; otherwise the fitness value is set to the total bandwidth used for mapping VN links, and the best position is selected as the one with the minimum fitness value. Furthermore, three factors are used to adjust the direction of particles, i.e., best position reached by particle, best position reached amongst all particles, and current inertial trend of particle. Finally, after a pre-defined maximum number of iterations, the best position reached across all the particles is selected as the optimal VNE solution. This particle swarm method is compared to the D-ViNE solution [NC01] via simulation, and results shows that it achieves higher average revenues and acceptance ratios.

Meanwhile [GS01] uses genetic algorithms to solve the VNE problem. Namely, a

"chromosome" is defined as a *binary* sequence with total length equal to the number of substrate network nodes. Elements in this sequence with a value of unity are then used to denote the corresponding substrate nodes that are assigned to VN nodes, i.e., VN mappings. The scheme then initializes a given number of chromosomes and computes a VN link mapping for each by solving a *mixed integer programming* (MIP) problem. In particular, the objective here is to maximize the link revenue, and the objective value for each chromosome is recorded. Subsequently, two child chromosomes are generated for each existing chromosome, and VN link mappings are computed for each in the same manner as above. Hence by iteratively generating child chromosomes, the scheme selects the one with the maximum link revenue as the optimal VNE solution. The authors compare their algorithm with a baseline VNE scheme similar to the one in [MY01], and results show higher revenues and lower transmission delays for VN links.

Finally, an ant colony optimization technique for VNE is also presented in [IF02]. In particular, this algorithm starts by dividing the VN topology into smaller components, i.e., star topologies. A set of artificial ants are then introduced to map all components (star topologies) in parallel, and here each ant repeats the mapping $N$ times over, i.e., $N$ iterations. In addition, the iterations for all ants are synchronized, i.e., each iteration is processed at the same time in parallel. Now in order to map each component, i.e., a VN node with adjacent VN links, a candidate substrate node list is built by selecting the substrate nodes close to adjacently-mapped VN nodes. A substrate node is then selected from this list by computing a probability value that is related to the available node/link resources at the node as well as the minimum bandwidth used to map this VN node in previous iterations. The latter provision tries to achieve more globalized selection. Finally, the mapping with the minimum bandwidth consumption level is selected as the optimal solution. The proposed solution is simulated and compared to the schemes in [YZ01] [MY01], and results show lower rejection rates and higher revenues/lower costs.

### 2.1.3 Graph-Based Heuristic Solutions

A number of heuristic schemes have also been developed to address VNE scalability concerns. These strategies use graph-based algorithms and are well-suited to adaptation in realistic scenarios, i.e., "on-demand" provisioning. Now in general, graph-based VNE heuristics can be classified into two key categories, i.e., separate node/link mapping (two-stage) and joint node/link mapping (single-stage). These types are now detailed further.

Two-stage VN mapping algorithms first map a subset of the VN nodes using various strategies and then compute their interconnecting VN links. For example, [YZ01] selects a cluster center by considering the "stress" levels of a node and its adjacent links. Here stress is measured as the number of VN nodes (links) that have been mapped onto the particular substrate node (link). The remaining VN nodes are then selected according to their node stress levels *and* distances to already-assigned nodes (where distance is defined as the shortest-distance path with link weights set proportional to link stress). Namely, an unassigned VN node is mapped in accordance with its node degree. Meanwhile the two-stage scheme in [MY01] first selects a set of candidate substrate nodes and then maps each VN node to an available substrate node with the maximum resource levels in this set. Next, these two-stage schemes proceed to route network connections for the requested VN links, with most using shortest-path [YZ01] or $k$-shortest path [MY01] algorithms. However, MCF routing schemes have also been used to route VN link connections in cases where path-splitting is supported at the substrate network [MY01]. Overall, simulation results show that incorporating node/link stress values gives much better load balancing, i.e., reducing the number of congested substrate nodes/links. Furthermore, [MY01] also shows that path-splitting increases average revenues and reduces average costs, i.e., versus no path-splitting.

Conversely, single-stage mapping schemes jointly map VN nodes and all their adjacent VN links, i.e., if and only if the adjacent VN nodes at the other end of the VN link have already been mapped [JL01], [XC01]. In general, these schemes can improve resource efficiency and also lower blocking. Along these lines, [JL01] presents a single-stage mapping algorithm based upon sub-graph isomorphism detection (using a backtracking search). Namely, this algorithm jointly maps a VN node along with its adjacent VN links, and if the mapping is infeasible, the scheme backtracks and re-maps the last VN node to another substrate node. Meanwhile, [HY05] presents another scheme that maps VN nodes in descending order of node degree. Namely, a candidate substrate node is selected for each VN node by considering its nodal resources as well as the communications cost from its location to the locations for adjacently-allocated VN nodes. After a VN node has been mapped, its adjacent VN link connections are then routed using a shortest-path algorithm, i.e., before the next VN node is mapped. In both [JL01] and [XC01], the authors compare their proposed schemes with the two-stage scheme in [MY01], and overall findings show significant advantages with single-stage mapping, i.e., in terms of VNE metrics such as revenue, cost and acceptance rates. Furthermore, simulations in [HY05] also show improved (lower) cost values than the scheme in [JL01].

## 2.2 Survivable VN Embedding Design

As more and more organizations migrate to the cloud, the reliability of underlying VN mappings is becoming a major concern. Namely, the failure of network substrate nodes or links can easily disrupt numerous VN mappings, particularly across higher-bandwidth substrates. In turn, these disruptions can lead to key breaches in client SLAs, incurring further economic penalties and revenue losses for operators. Hence, many providers are very interested in *survivable* VNE schemes to handle different

failure scenarios at the network substrate layer. Along these lines, various solutions have been proposed for a range of failure scenarios, see also Figure 2.2. Consider the details.

### 2.2.1 Single Link Failure Protection Schemes

The authors in [MR01] propose a hybrid policy heuristic to handle single link failures for VN mappings. Basically, this scheme pre-partitions the bandwidth in each substrate link into two parts, i.e., one for working VN links and the other for (potential) post-fault restoration support. Now the scheme starts by pre-computing a set of detour routes for each physical substrate link using any adaptable path selection algorithm, e.g., $k$-shortest path. Next, VN node mapping is done for each arriving VN request by using the existing heuristics in [YZ01], [NC01]. After all the requested VN nodes have been mapped, the VN links are computed by solving a MCF problem, i.e., path splitting is allowed. Now if/when a single link fails, all of its traversing flows are re-directed onto the set of pre-computed detours. However, recovery guarantees still cannot be provided here for all interrupted flows, i.e., due to limited reserved capacity on detour routes. Instead, the goal is to try to minimize the amount of failed (unrecovered) bandwidth. Overall, this proposed scheme is compared to two baseline methods, i.e., one which re-computes a new mapping for an affected VN mapping and another which pre-reserves backup bandwidth for each VN link. The results here show that the proposed solution gives higher acceptance ratios and operating profit for carriers.

The authors in [YC01] also propose a survivable VNE scheme for single link failures called the *pardalis* algorithm. This solution uses a VN mapping algorithm similar to that in [YZ01] to first compute a working VN mapping. Namely, an available resource value, $\gamma(n_s)$, is first defined for substrate node $n_s$ as the residual

node resource level multiplied by the sum of residual bandwidth of all adjacent substrate links. A direct resource constraint, $\gamma(n_v)$, is also defined for a VN node $n_v$ as the requested node resource level multiplied by the sum of requested bandwidths of all adjacent VN links. The scheme then maps VN nodes with larger $\gamma(n_v)$ values to substrate nodes with larger $\gamma(n_s)$ values. VN links are then routed using shortest path algorithms. Now in order to provide resiliency, backup protection paths are also computed for the VN links, i.e., with further resource sharing between protection routes with link-disjoint primary (working) routes. Specifically, backup connections are routed by pruning working connection VN links and using dynamic weights to efficiently share/re-use allocated backup capacity. The pardalis algorithm is then compared to several baseline variants, i.e., those not incorporating $\gamma(n_s)$ and $\gamma(n_v)$ in the VN mapping, those not implementing resource sharing along protection paths, etc. Overall results show lower costs and protection bandwidth overheads with the proposed scheme.

[XZ01] also presents another scheme for single-link failure recovery using backup VN links. Namely, the scheme first prunes the substrate physical topology to build a reduced logical topology containing a subset of the original physical nodes and paths between these subset nodes, i.e., logical link can consist of physical path computed using any shortest-path algorithm. Here, this logical topology is created using *quality of service* (QoS) constraints, i.e., such as link delays. Next, an ILP formulation is proposed to map the VN nodes (links) onto this logical topology, with each VN link having a pair of disjoint working and backup paths. However, this approach has several shortcomings. First, since logical links are computed using a heuristic algorithm, the solution of the ILP is not optimal in any sense. Second, the scheme does not take into account link and node resource constraints in the VN mappings. Finally, the ILP formulation tries to minimize the total number of nodes used for backup paths. However, this is not a really major concern as these nodes are only acting as intermediaries along a path and do not require nodal resources. Now

owing to the complexity of the ILP model, the authors also propose a resilient VN mapping heuristic based upon the same assumptions in [XZ02]. In particular, this scheme iteratively maps each VN node (onto the logical topology) and finds two link-disjoint paths for its adjacent VN links. Although this heuristic is more efficient than the ILP, it still has its limitations as in [XZ01], i.e., pre-computed logical links limit potential solutions, lack of bandwidth and nodal resource constraints, etc.

Meanwhile the authors in [TG01] also propose a resilient VNE scheme for single link failures. This framework supports path splitting for VN links and pursues resilience at the substrate *link* level. Namely, a set of bypass paths are defined for each substrate link, and all affected flows on the failed link are redirected along these bypass paths. Note that these path sets (between two substrate nodes) are pre-computed using a $k$-shortest-path algorithm and then used to map the VN links *and* protect the substrate links. Using this framework, two VNE recovery schemes are defined here, i.e., *shared on-demand* (SOD_BK) and *shared pre-allocation* (SPA_BK). Namely, SOD_BK re-uses any existing VN node mapping algorithm to first map the VN nodes. An LP formulation is then proposed to route the VN links, i.e., with protection being achieved by the pre-computed bypass paths for each substrate links. Now owing to the single link failure assumption, resource sharing can also be done between protection flows on bypass paths. Overall, SOD_BK allocates backup link resources after a VN request arrives. Conversely, SPA_BK pre-allocates backup link resources at the initial stage before any VN request arrives. Namely, an LP formulation is first used to divide the bandwidth on each substrate link into two parts, i.e., one for working VN link mappings and the other for protection mappings. The goal here is to maximize the total bandwidth allocation for working mappings, and link resource sharing is also applied (due to the single link failure assumption). Detailed simulations show that both the SOD_BK and SPA_BK schemes achieve higher acceptance ratios (lower blocking) and revenues than schemes without link resource sharing, i.e., since fewer backup resources are used.

The work in [XY01] also proposes a survivable mapping scheme for user *application specific and agile private* (ASAP) topologies. Namely, an ASAP topology is formed by a set of tasks and communications between tasks, and is similar to a VN request. Also, the failure model assumes a single substrate node failure along with a (concurrent) substrate link failure. Overall, two survivable schemes are proposed here, i.e., *cluster and path protection* (CPP) and *virtual network protection* (VNP). The former computes a backup ASAP network mapping for the primary mapping, and in both mappings, two disjoint paths are computed for each tasks communication link (VN link). The purpose of this backup mapping is to recover from node failures and improve link resilience. Link bandwidth sharing is also done between the protection paths. Meanwhile, the VNP scheme computes three topology-disjoint mappings for an ASAP network request since two failures (node and link) may occur in the substrate. Detailed simulation results show that the CPP scheme incurs less cost than the VNP scheme if node resources are more expensive than link bandwidth resources.

The authors in [HY01] and [HY06] also propose a novel survivable VNE solution for single link failures, termed as the *migratory protection-based virtual infrastructure mapping algorithm* (MP-SVIMA) scheme. This strategy is different from traditional link protection algorithms which utilize a pair of link-disjoint paths. Instead, backup nodes are pre-reserved and migration is only done for one end-point of the affected VN link, i.e., in order to protect from substrate link failures. Hence in addition to node resources for backup nodes, backup paths are also needed from the (new) backup nodes of the neighboring VN nodes, termed here as *migratory backup path groups*. The scheme also allows bandwidth sharing between the original working path and its corresponding migratory backup path group, i.e., called *intra-share*. In addition, link bandwidth and node resources can also be shared between migratory backup path groups, i.e., called *inter-share*. Now in order to implement MP-SVIMA, a traditional link-disjoint protection strategy is defined, termed as the *protection-based survivable*

*virtual infrastructure mapping algorithm* (P-SVIMA). This algorithm re-uses the P-SVIMA scheme to first compute a basic link-disjoint survivable VN mapping. Next, for each VN link, MP-SVIMA computes the cost of using migratory protection to protect the link. If this cost is less than that for basic link-disjoint protection, then the migratory protection mapping is adopted for this VN link. Otherwise, the original link-disjoint protection solution is kept. Overall, simulations show that the MP-SVIMA scheme uses fewer backup resources, lowers costs, and also improves blocking ratios as compared to the P-SVIMA scheme.

Finally, [BG01] presents another survivable VNE scheme. Akin to the work in [HY01], this approach also supports VN node migration to overcome single link failures. However, after a link failure happens, all VN nodes are allowed to switch their locations, i.e., instead of just one VN node as in [HY01]. In addition, the scheme also restricts added backup nodes and only allows redundant additional bandwidth resources in the substrate network. The goal here is to minimize the total backup bandwidth used. Now this scheme can work with (re-use) any existing VNE algorithm to compute an initial working VN mapping. Next, for each substrate link failure, the scheme computes a backup mapping using the minimum amount of additional resources. This is done by removing the failed substrate link and then iterating over all VN nodes to see if their mappings can be improved. Namely, the new VN mappings are computed by keeping the VN node fixed or switching it with all other VN nodes. From these VN mappings, the one with the minimum additional cost is selected. Finally, after all iterations are complete, the resulting VN mapping is selected as the protection mapping for this substrate link failure. Overall, simulation results show that the proposed scheme achieves lower redundant link costs and blocking ratios versus traditional link-disjoint pair protection schemes. However, the tradeoff here is increased VN node migration costs.

### 2.2.2 Single Node Failure Protection Schemes

Node-level failures are also a key concern in cloud environments, i.e., switching nodes, data-center sites. As a result, [HY02] and [HY03] propose a survivable VNE schemes for such scenarios. However, the network model here is slightly different and assumes that a substrate node is divided into two parts, i.e., a facility node and a switching node. Namely, a facility node represents data-center resources (such as computation and storage) and is directly connected to the switching node which links to the substrate network. Therefore, VN nodes can only be mapped to facility nodes. The work also assumes that failures only affect facility nodes and not underlying switching nodes or links (somewhat restrictive). In addition, only a subset of the VN nodes are deemed as critical and need to be protected. Based upon these assumptions, two VNE recovery solutions are proposed, i.e., *1-redundant* and *K-redundant* schemes. In both cases, a redundant VN graph is created first. Now the former scheme uses one redundant VN node to protect all critical VN nodes. Conversely, the latter scheme uses $K$ redundant VN nodes to protect the critical VN nodes, i.e., where $K$ is equal to the number of critical VN nodes. Furthermore, in both cases (additional) redundant VN links are also needed to connect the redundant VN nodes to all the neighboring VN nodes of the protected VN node. Hence, after this redundant VN graph is built, it is mapped onto the substrate network. Now in order to efficiently utilize resources, two sharing strategies are also applied here, i.e., *cross share and backup share*. The former approach also allows backup paths (connecting to a redundant VN node) to share resources with the original working paths they protect. Meanwhile, the latter approach allows sharing between backup paths that protect different VN nodes, i.e., since only one facility node can fail at a given time. In addition, the $K$ redundant VN nodes (in the $K$-redundant solution) can be mapped to the same set of substrate nodes, i.e., such that only $k$ ($1 \leq k \leq K$) substrate nodes are used for backup support. Now in order to solve this survivable shared VN mapping problem,

a MILP formulation is presented. However, as this problem is intractable, the D-ViNE algorithm from [NC01] is instead used to compute the working mapping first. A simplified MILP is then solved to get the final backup solution. Overall results confirm that resource sharing greatly lowers the cost of providing VN survivability. In addition, the $K$-redundant solution is also much more efficient than the 1-redundant solution when substrate link costs are higher than substrate node costs.

Meanwhile [CQ01] presents another survivable VN scheme for single facility node failures. Akin to [HY02], this approach also uses one redundant VN node to build a redundant VN graph, called an *enhanced VN* (EVN). However, unlike the 1-redundant solution [HY02], it also allows the unaffected VN nodes to be remapped in the EVN topology, i.e., called *failure-dependent EVN* (FD-EVN). In particular, for a VN request with $N$ VN nodes, the scheme defines a $N + 1$ "nodes edit grid" to first construct the EVN. Namely this grid is a fully-connected graph between these $N + 1$ nodes. Now for each potential node failure in the edit grid [DJ01], the scheme re-maps the VN topology to the residual part of the edit grid to avoid any failure overlaps. Sharing is then done to combine all the allocated resources for each of the above-computed VN topologies, thereby yielding an EVN that can recover from any arbitrary node failure. However, since the optimization solution is very intractable, the authors also propose a heuristic scheme using graph transformation/decomposition and bipartite graph matching [HK01]. In particular, a mapping cost matrix is computed for each node failure in the edit grid, i.e., where each matrix element, $s_{ij}$, represents the cost of migrating VN node $i$ (and of its all adjacent VN links) to an edit grid node $j$. An optimal solution is then selected using this mapping cost matrix. Now since the main focus here is to construct an EVN, the authors simply re-use existing VN mapping algorithms to map the EVN onto the substrate network. Simulation results show that the proposed FD-EVN scheme achieves lower resource consumption and higher acceptance ratios (lower blocking) than several other schemes. However, the tradeoff here is in terms of increased numbers of node

migrations after failure events, i.e., increased operational complexity.

### 2.2.3 Multi-Failure Protection Schemes

Multi-failure recovery is also a major concern given the devastating impacts of large-scale stressors such as natural disasters, malicious attacks, cascading power outages, etc. However, only a handful of schemes have looked at such scenarios in the VNE context. For example, [WY01] proposes a solution for protecting critical VN nodes under multiple nodes failures. First of all, this work assumes that each node has a failure probability, and that all nodes fail in an independent manner. Under this assumption, the predefined reliability guarantee level, $r$, of a VN (with $n$ critical nodes) can be achieved by providing $k$ backup VN nodes. Since the relationship between $k$ and $r$ is non-linear, backup node pooling is also used to improve resource efficiency, i.e., multiple VN requests can share backup VN nodes. Now backup VN links are also needed for multi-failure recovery. Therefore, the scheme divides the backup VN links into two types, i.e., those connecting backup nodes to all neighbors of protected nodes, and those interconnecting backup nodes. However, as these backup links will not be used at the same time, link resource sharing can be done here. Finally, a MILP formulation is also proposed to map this resilient VN topology onto a substrate network with the objective of minimizing resource usage. Simulations are then done to compare the performance of the scheme with a modified version without backup node pooling and backup link resource sharing. Overall results show that node/link sharing can greatly lower both blocking ratios and resource usages. However this work assumes that a substrate link and its end nodes will not fail, an unrealistic assumption.

The more challenging case of multiple substrate node/link failures is also studied in [HY04], i.e., for large-scale regional stressors/disaster events. In particular,

multiple backup VN mappings are computed for each potential failure region (which in turn is defined as a sub-graph of vulnerable nodes/links). However, since it is assumed that only a single regional stressor can occur at a given time, resource sharing is also done between all backup VN mappings (and their working VN mapping). A MILP formulation is then proposed to minimize resource usages across all VN mappings. However, owing to excessive computational complexity here, this optimization is not solved and instead a heuristic *separate optimization with unconstrained mapping* (SOUM) scheme is proposed. This scheme basically allows full resource sharing between the backup and primary VN mappings. Another variant, termed as the *incremental optimization with constrained mapping* (IOCM) scheme, is also proposed. Here, a working mapping is first computed, and then additional backup node/bandwidth resources are incrementally added to handle each failure region, i.e., backup resources are only added for nodes affected by a given failure. Simulations show that the SOUM scheme gives better blocking reduction but also yields higher costs and increased number of post-fault VN node migrations. Meanwhile, the work in [GS02] actually solves the MILP formulation in [HY04] by using Lagrangian relaxation and decomposition-based techniques. In both methods, the original MILP formulation is decomposed into $R$ sub-problems to simplify the computational complexity, where $R$ is the number of failure regions. Overall finding indicate that both the Lagrangian relaxation and decomposition-based schemes can achieve the same mapping costs as the full MILP, but with much faster (scalable) run times.

## 2.3 Open Challenges

Overall, VN survivability is an important focus area that has seen notable focus in recent years. However many further problems and challenges remain and need

to be explored. Foremost, the design of efficient multi-failure VN recovery schemes is a key concern, as only a few initial solutions have been proposed here. In particular, these existing schemes are rather resource-intensive and provision backup VNs to handle all possible stressor cases. Furthermore, these strategies tend to pursue an "all-or-nothing" protection approach by provisioning full recovery. Indeed, much better efficiencies can be achieved by incorporating the probabilistic nature of large-scale stressor events. However, there are no known studies on probabilistic VN recovery. Finally, post-fault restoration offers a very viable "last-gap" recovery solution in case pre-provisioned VN protection fails. However, this approach has not been studied within the context of VN recovery. Along these lines, post-fault restoration schemes can be developed to perform VN node and link remapping after large stressor events, and these solutions can draw from a wide range of existing heuristic strategies. Overall, these open problems form the main motivations for this dissertation research.

# Chapter 3

# Multi-Failure VN Protection

This chapter studies pre-provisioned protection for multi-failure VN recovery. Now as surveyed in Section 2.2.3, only a handful of related schemes have been proposed here. For example, some solutions provision backup VN nodes (along with their necessary backup VN links) to protect against multiple substrate node failures [WY01]. However these algorithms do not handle substrate link failures nor do they incorporate the geographical patterns of multi-failure stressors, i.e., instead assuming arbitrary concurrent node failures. Alternatively, more recent efforts have looked at VNE provisioning for geographically-correlated failures [HY04],[GS02]. However, these schemes provision backup resources for *each* potential failure region and therefore yield very high costs/blocking.

In light of the above, this chapter presents a novel VN survivability scheme for handling multiple region-based failures with improved resource efficiency. Overall, the work leverages a similar philosophy to that used in earlier studies on *shared risk link group* (SRLG) protection for regular *point-to-point* (P2P) connections [EO01]. Namely, working and protection VN mappings are computed for each request so as to ensure that they are failure region-disjoint, i.e., to guarantee recovery from a

single regional failure event. To achieve this, a MILP formulation is first proposed by extending the VN model in [NC01] with new constraints to prevent primary/backup mappings from traversing the same failure regions. However, owing to high MILP complexity, two heuristic schemes are also proposed here. Namely, the *failure region group based mapping* (FRGBM) scheme divides the failure regions into two fixed groups and only allows a working mapping to use resources in one of these groups. Meanwhile, the *dynamic failure region disjoint mapping* (DFRDM) scheme computes VN mappings without performing prior separation of failure regions. Since both of these solutions only provision two VN mappings, they can achieve lower resource utilization and (blocking) versus other survivable VN strategies. Complete details are now presented.

## 3.1 Network Model and Description

Before detailing the MILP formulation and related heuristic strategies, the overall network model for VN mapping is presented along with the requisite notation.

### 3.1.1 Substrate Network

The substrate network is modeled as an undirected graph $G_s = (V_s, E_s)$, where $V_s = \{v_s^1, v_s^2, ..., v_s^{|V_s|}\}$ is the set of substrate nodes and $E_s = \{(v_s^i, v_s^j)|v_s^i, v_s^j \in V_s\}$ is the set of substrate links. Here each substrate link $e_s \in E_s$ has a fixed bandwidth capacity and each substrate node $v_s \in V_s$ has a fixed amount of computing and storage resources. Now in order to simplify the discussions, the available bandwidth of a physical substrate link $e_s \in E_s$ is given by $B(e_s)$ and its unit bandwidth cost by $\mathbb{C}(e_s)$. Similarly, the node resource capacity for a substrate node $v_s \in V_s$ is also given by $R(v_s)$ and its unit cost by $\mathbb{C}(v_s)$. Additionally, substrate link $e_s$ is also denoted

as $(v_s, v'_s)$, where $v_s$ and $v'_s$ are the two link node end-points. Overall, Figure 3.1a shows a sample 10-node substrate network hosting a 3-node VN request, where the numbers next to the links (nodes) represent the available bandwidth (node) resources levels.

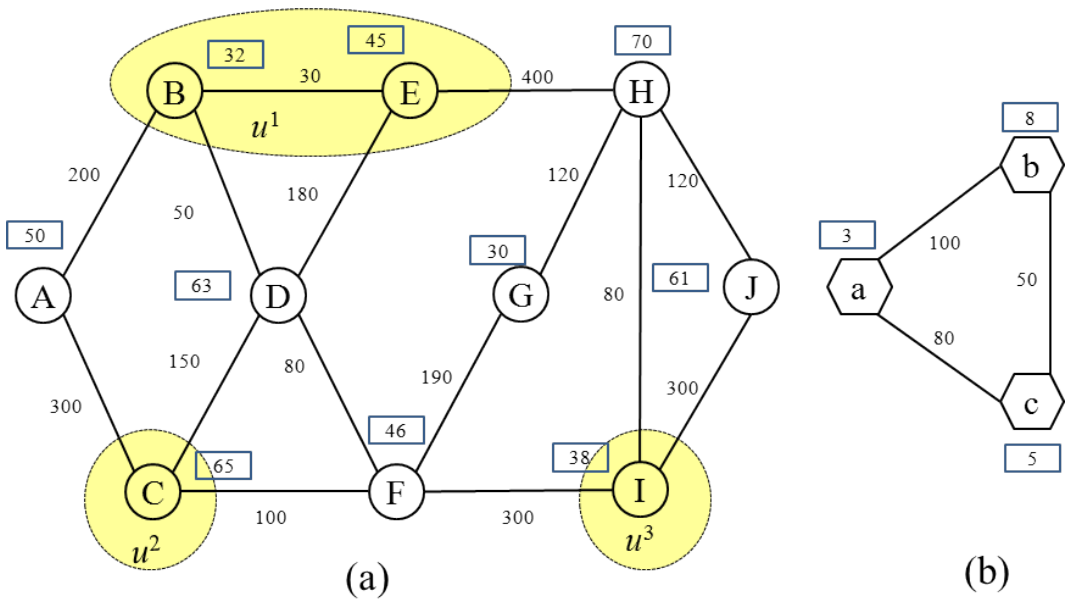



Figure 3.1: (a) substrate network, (b) sample VN request

### 3.1.2 VN Request

A VN request is given by an undirected graph $G_v = (V_v, E_v)$, where $V_v$ is the set of VN nodes and $E_v$ is the set of VN links. Here each VN node $v_v \in V_v$ requires a certain amount of nodal resources, denoted as $r(v_v)$, and each VN link $e_v \in E_v$ also requires a given bandwidth, denoted as $b(e_v)$. Similar to a substrate link, a VN link $e_v$ is also denoted by $(v_v, v'_v)$. Furthermore, a VN node-to-substrate node mapping is denoted as $< v_v, v_s >$, i.e., VN node $v_v$ mapped to substrate node $v_s$. Again, Figure 3.1b shows a sample VN request with associated nodal resource and link bandwidth

requirements.

### 3.1.3 Regional Failure Model

Regional failures are usually characterized by highly-correlated node and link outages, i.e., centered about a specific geographic region and ocurring at or close to a particular instance in time. Along these line, the proposed failure model incorporates a finite number of failure regions (stressors) in a set $U$. Furthermore each potential stressor, $u \in U$, is defined by a sub-graph $G_u = (V_u, E_u)$, where $V_u \subseteq V_s$ is the set of substrate nodes affected by the event and $E_u \subseteq E_s$ is the set of substrate links affected by the event.

Now it is generally safe to assume that all events in $U$ are independent and mutually-exclusive, i.e., only one can occur at a given time [HL01]. Given these assumptions, it can also be assumed that all events are non-overlapping in terms of substrate nodes, i.e., $v_s \notin V_{u^j}, \forall v_s \in V_{u^i}, \forall i \neq j$. Note that this is not an overly-restrictive requirement per say, as any overlaping substrate nodes between two/more failure regions can be separated out and placed in a new failure region. An example of this notation is shown in Figure 3.2 for a 24-node network with 5 failure regions, i.e., $U = \{u^1, u^2, ..., u^5\}$ and $G_{u^1}(V_{u^1}, E_{u^1})$, where $V_{u^1} = \{v_s^2, v_s^3, v_s^4\}$ and $E_{v^1} = \{(v_s^1, v_s^2), (v_s^2, v_s^3), (v_s^2, v_s^4), (v_s^2, v_s^6), (v_s^3, v_s^4), (v_s^3, v_s^6), (v_s^4, v_s^7)\}$.

### 3.1.4 Performance Evaluation Metrics

Some of the key metrics used to study VN performance are now presented. Foremost, operators provisioning VN services will generally want to achieve a high level of resource efficiency over their underlying substrate networks. In addition, associated revenue generation (cost reduction) concerns will also be very important here [GS01],

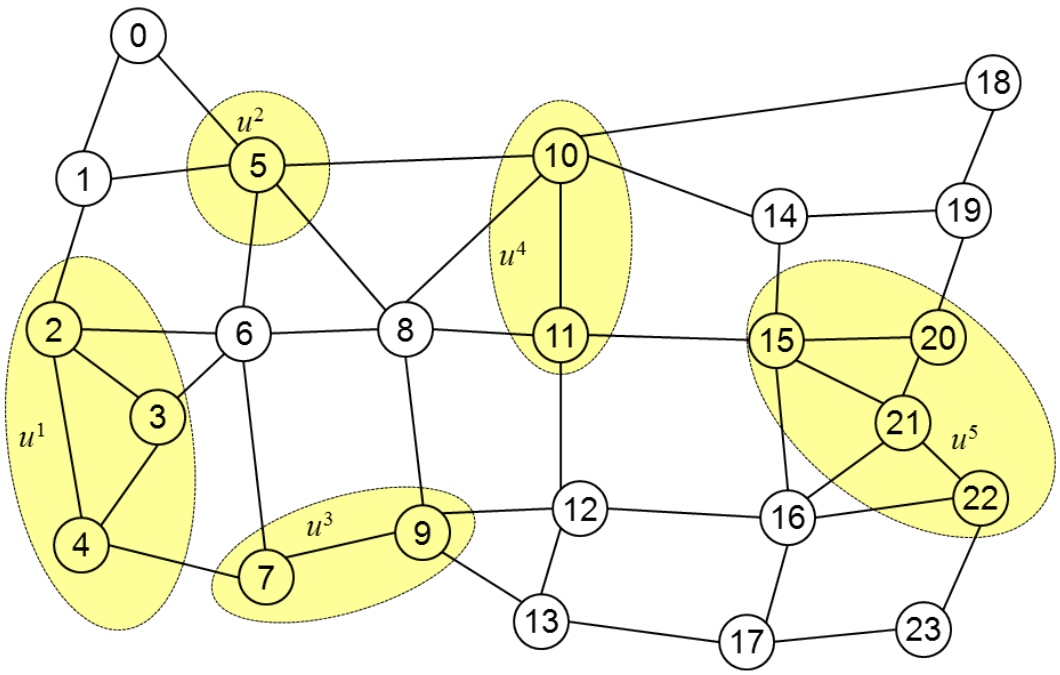



Figure 3.2: 24-node substrate network with 5 failure regions, $|U|$=5

[XC01]. As a result the revenue associated with provisioning a VN request is defined using the formulation in [XC01] as:

$$REV(G_v) = \sum_{e_v \in E_v} b(e_v) * \mathbb{I}(e_v) + \rho \sum_{v_v \in V_v} r(v_v) * \mathbb{I}(v_v) \qquad \text{(Eq. 3-1)}$$

where $\rho$ is the fraction of nodal resource revenue, $\mathbb{I}(e_v)$ is the revenue per unit of bandwidth, and $\mathbb{I}(v_v)$ is the revenue per unit of nodal resource. Furthermore the cost of accepting a VN is also given by [XC01]:

$$COST(G_v) = \sum_{e_s \in E_s} \mathcal{F}_{e_s}^{G_v} * \mathbb{C}(e_s) + \pi \sum_{v_s \in V_s} \mathcal{N}_{v_s}^{G_v} * \mathbb{C}(v_s) \qquad \text{(Eq. 3-2)}$$

where $\pi$ is the fraction of nodal resource cost, $\mathcal{F}_{e_s}^{G_v}$ is the total amount of bandwidth allocated on substrate link $e_s$ for mapping the VN, and $\mathcal{N}_{v_s}^{G_v}$ is the total amount of nodal resources allocated on the substrate node $v_s$ for mapping the VN (and $\mathbb{C}(e_s)$ and $\mathbb{C}(v_s)$ are introduced in Section 3.1.1).

Carefully note that this cost formulation is different from the one defined for non-survivable VN mappings in [XC01], i.e., since the link bandwidth (node) resource costs for a substrate link (node) in Eq. 3-2 are not directly related to a VN link (node). Specifically, survivable VN mappings also require backup link bandwidth (node) resources and usually implement resource sharing. Hence the total amount of link bandwidth (node) resources allocated on the substrate links (nodes) can only be determined after the survivable mapping has been computed.

Now from an operator's perspective, it is very desirable to increase long-term revenue. Along these lines, this value is defined here as:

$$\frac{\sum_i REV(G_v^i)}{T}, \forall G_v^i \in \mathbb{A} \qquad \text{(Eq. 3-3)}$$

where $G_v^i$ is the *i-th* VN request, $\mathbb{A}$ is the set of all accepted VN requests, and $T$ is the total running time. Similarly, the long-term average cost is defined as:

$$\frac{\sum_i COST(G_v^i)}{T}, \forall G_v^i \in \mathbb{A} \qquad \text{(Eq. 3-4)}$$

However, to precisely describe the operator profit, the *net* revenue is computed as follows:

$$\frac{\sum_i (REV(G_v^i) - COST(G_v^i))}{T}, \forall G_v^i \in \mathbb{A} \qquad \text{(Eq. 3-5)}$$

Note that other metrics can also be considered here. For example, VN request blocking rates can provide a key measure of lost revenue, and hence many operators will want to minimize these values to improve their performance. Additionally, protection switching overheads after failures (to backup VN nodes and links) can also be considered in order to gauge the level of user service disruptions.

## 3.2 MILP Formulation

A detailed optimization formulation for survivable VN mapping is now presented based upon the above notation. This approach pursues a single-objective function

to minimize overall resource usage for a VN request.

### 3.2.1 Augmented Substrate Graph Model

In order to formulate the survivable VN mapping problem, an augmented substrate graph $G_a(V_a, E_a)$ is first constructed, akin to [NC01],[HY04]. Namely, a meta-node, $v_m$, is first created for each VN node, $v_v$, and then connected to all underlying substrate nodes, $v_s$, via meta-edges, $e_m$, with infinite bandwidth. Hence this augmented substrate graph is a combination of the original substrate graph and the meta-nodes/meta-edges, i.e., $V_a = V_s \cup V_m$, $E_a = E_s \cup E_m$, where $E_m = \{(v_m, v_s) | v_m \in V_m, v_s \in V_s\}$ is the set of meta-links. It is also assumed that regional failures do not affect any of the meta-nodes since they are auxiliary nodes representing VN node mappings. However meta-edges can fail if their underlying connecting substrate nodes fail. An example of an augmented graph is shown in Figure 3.3 for a 5-node substrate network with 2 VN nodes.

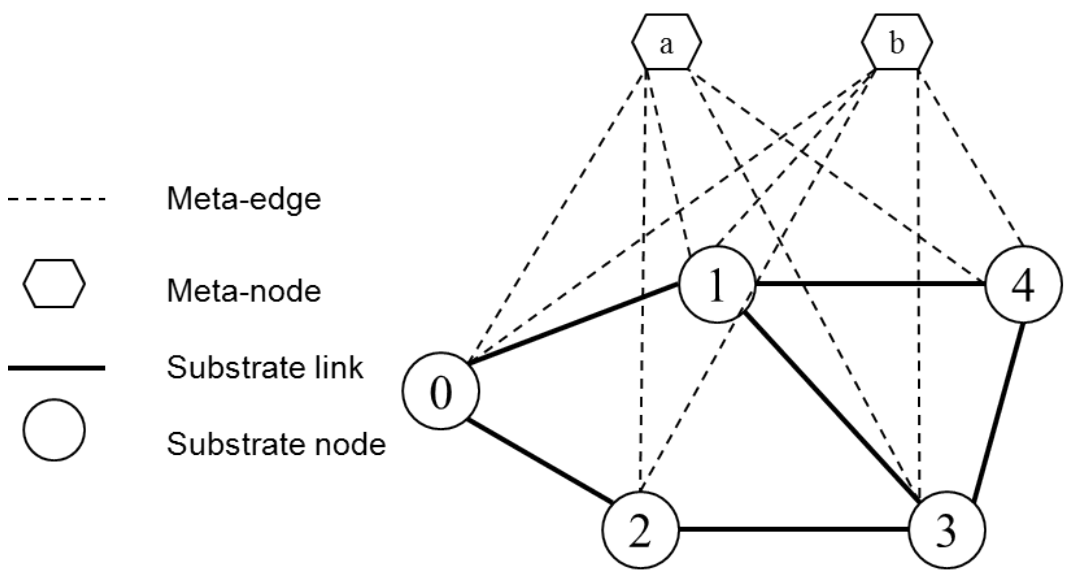



Figure 3.3: Example of an augmented graph with meta-nodes/meta-edges

### 3.2.2 Failure Region Disjoint MILP Formulation

The failure region-disjoint survivable VN mapping is formulated as a mixed integer multi-commodity flow problem. Namely, each VN link $e_v$ is treated as a commodity with source and destination nodes $s \in V_m$ and $t \in V_m$, i.e., flows start and end at distinct meta-nodes. By further restricting the meta-edges, each meta-node can be forced to choose a single meta-edge to connect to the substrate network. This effectively selects a substrate node for each meta-node corresponding to its mapping. At the same time, VN links (connections) must also be mapped on to substrate network links. Now since the goal is to generate failure region-disjoint working and protection mappings, added restrictions are placed to ensure that only one mapping is assigned to a failure region $u \in U$. Specifically, this is done by using the index variable $z \in Z = \{1, 2\}$ to denote the mapping, i.e., $z = 1$ for working and $z = 2$ for protection. As per the formulation, the following variables are defined.

**Variables:**

- $f_{mn}^{q,z}$: Flow variable denoting the total amount of flow from node $m$ to node $n$ on the substrate edge $(m, n) \in E_a$ for the VN link $q \in E_v$ of mapping $z \in Z$.

- $\alpha_{mn}^{z}$: Binary variable which is 1 if the flow in any of the VN links in mapping $z$ uses the substrate edge $(m, n)$, i.e., $\sum_{q \in E_v}(f_{mn}^{q,z} + f_{nm}^{q,z}) > 0$; otherwise it is 0.

- $\varepsilon_{u}^{z}$: Binary variable which is 1 if the mapping $z$ can be mapped in failure region $u \in U$; otherwise it is 0.

- $b_e$: Variable denoting the maximal amount of flow on the substrate link $e \in E_s$.

- $r_n$: Variable denoting the maximal amount of node resource allocated from the substrate node $n \in V_s$.

Using the above definitions, the following objective function and constraint equations are defined.

**Objective:**

$$min \sum_{e \in E_s} b_e * \mathbb{C}(e) + \sum_{n \in V_s} r_n * \mathbb{C}(n) \quad \text{(Eq. 3-6)}$$

**Constraints:**

$$\sum_{z \in Z} \varepsilon_u^z \leq 1, \forall u \in U \quad \text{(Eq. 3-7)}$$

$$\alpha_{mn}^z \leq \varepsilon_u^z, \forall (m,n) \in E_u, \forall z \in Z, \forall u \in U \quad \text{(Eq. 3-8)}$$

$$\sum_{n \in V_a} \alpha_{mn}^z \leq A * \varepsilon_u^z, \forall m \in V_u, \forall z \in Z, \forall u \in U \quad \text{(Eq. 3-9)}$$

$$\sum_{q \in E_v} (f_{mn}^{q,z} + f_{nm}^{q,z}) \leq B(m,n) * \alpha_{mn}^z, \forall (m,n) \in E_a, \forall z \in Z \quad \text{(Eq. 3-10)}$$

$$r(m) * \alpha_{mn}^z \leq R(n), \forall m \in V_m, \forall n \in V_s, \forall z \in Z \quad \text{(Eq. 3-11)}$$

$$\sum_{n \in V_a} f_{mn}^{q,z} - \sum_{n \in V_a} f_{nm}^{q,z} = 0, \forall q \in E_v, \forall m \in V_a \backslash \{s_q, t_q\}, \forall z \in Z \quad \text{(Eq. 3-12)}$$

$$\sum_{n \in V_a} f_{s_q n}^{q,z} - \sum_{n \in V_a} f_{n s_q}^{q,z} = b(q), \forall q \in E_v, \forall z \in Z \quad \text{(Eq. 3-13)}$$

$$\sum_{n \in V_a} f_{t_q n}^{q,z} - \sum_{n \in V_a} f_{n t_q}^{q,z} = -b(q), \forall q \in E_v, \forall z \in Z \quad \text{(Eq. 3-14)}$$

$$\sum_{n \in V_s} \alpha_{mn}^z = 1, \forall m \in V_m, \forall z \in Z \quad \text{(Eq. 3-15)}$$

$$\sum_{m \in V_m} \alpha_{mn}^z \leq 1, \forall n \in V_s, \forall z \in Z \quad \text{(Eq. 3-16)}$$

$$\alpha_{mn}^z = \alpha_{mn}^z, \forall m,n \in V_a, \forall z \in Z \quad \text{(Eq. 3-17)}$$

$$\sum_{q \in E_v} (f_{mn}^{q,z} + f_{nm}^{q,z}) \leq b_e, \forall e \equiv (m,n) \in E_s, \forall z \in Z \quad \text{(Eq. 3-18)}$$

$$\sum_{m \in V_m} \alpha_{mn}^z * r(m) \leq r_n, \forall n \in V_s, \forall z \in Z \quad \text{(Eq. 3-19)}$$

$$f_{mn}^{q,z} \geq 0, \forall m,n \in V_a, \forall q \in E_v, \forall z \in Z \quad \text{(Eq. 3-20)}$$

$$\alpha_{mn}^{z} \in \{0,1\}, \forall m,n \in V_a, \forall z \in Z \quad \text{(Eq. 3-21)}$$

$$\varepsilon_{u}^{z} \in \{0,1\}, \forall z \in Z, \forall u \in U \quad \text{(Eq. 3-22)}$$

Overall, the objective function in Eq. 3-6 tries to minimize the cost of mapping a VN request (summed over all VN node and link mapping costs). Eq. 3-7 also constrains each failure region to be covered by at most one mapping. Meanwhile Eq. 3-8 pertains to link failures and ensures that if a link $(m,n)$ is located in a failure region $u$, then $\alpha_{mn}^{z}$ can only be set to 1 if this mapping $z$ can be mapped to failure region $u$. Similarly, Eq. 3-9 deals with node failures, an $A$ represents a large constant (set to greater than the maximal node degree in $G_a$). This constraint basically ensures that if a node $m$ is located in a failure region $u$, then its adjacent links can only carry flows if the whole mapping $z$ can be placed in failure region $u$. Next, Eq. 3-10 and Eq. 3-11 bound link capacity and nodal resources, respectively, i.e., since summing $f_{mn}^{q,z}$ and $f_{nm}^{q,z}$ ensures that the total flow in both directions of an undirected link $(m,n)$ is less than the available bandwidth. Meanwhile, Eq. 3-12, Eq. 3-13, and Eq. 3-14 implement flow conservation. Furthermore, Eq. 3-15 ensures that only one substrate node is selected for a meta-node, whereas Eq. 3-16 ensures that a substrate node can only be allocated to at most one meta-node. Next, Eq. 3-17 ensures that $\alpha_{mn}^{z}$ is the same in both link directions. Also, Eq. 3-18 and Eq. 3-19 restrict the maximum resource allocations for each substrate link/node in each mapping $z \in Z$. Finally, Eq. 3-20, Eq. 3-21 and Eq. 3-22 denote the necessary non-negative and binary constraints on $f_{mn}^{q,z}$, $\alpha_{mn}^{z}$, $\varepsilon_{u}^{z}$. Overall, this MILP model poses very high complexity. For example, mapping a 5-node/10-link VN request over a 20-node/40-link substrate network with 5 failure regions yields 6,230 variables.

## 3.3 Failure Region Disjoint Survivable VN Mapping

Owing to the high computational complexity of the above MILP formulation, some efficient heuristic-based algorithms are now proposed. In general, recovery schemes for single link/node failures are not very effective in multi-failure scenarios. Hence in order to achieve survivability under multiple faults, an alternate approach can be used to compute multiple VN mappings, with each avoiding a different failure region [HY04]. Furthermore, resource sharing can also be implemented between these mappings to help lower resource usages. This is shown more clearly in Figure 3.4, where VN request $G_v^1$ is assigned a working mapping $\{< a, A >, < b, B >, < c, C >\}$ and a protection mapping $\{< a', C >, < b', D >, < c', F >\}$. Now since substrate node $C$ is allocated to VN node $c$ in the working mapping and to VN node $a$ in the protection mapping, node resource sharing can be done here. In addition, since the link $(C, D)$ is used to route VN link connections for both mappings, bandwidth sharing can also be done here. Another sharing case is also seen for request $G_v^2$ where a VN node/link is mapped to the same substrate node/path in both the working mapping and protection mappings.

However, computing multiple backup mappings for each potential failure region $u \in U$ is clearly very resource intensive. As a result, a more resource-efficient survivable VN mapping solution is introduced here to compute two separate "failure-region disjoint" VN mappings for each incoming request, i.e., working and protection VN mappings. Now since the survivable VN mapping algorithm here is decomposed into two *non-survivable* mappings, any existing (regular) VN mapping scheme can be applied. Hence for the purposes of this study, the *non-survivable virtual infrastructure mapping* (NSVIM) algorithm from [HY04],[GS02] is adopted as the base VN mapping solution. Namely, this scheme uses a single-stage mapping to embed a VN

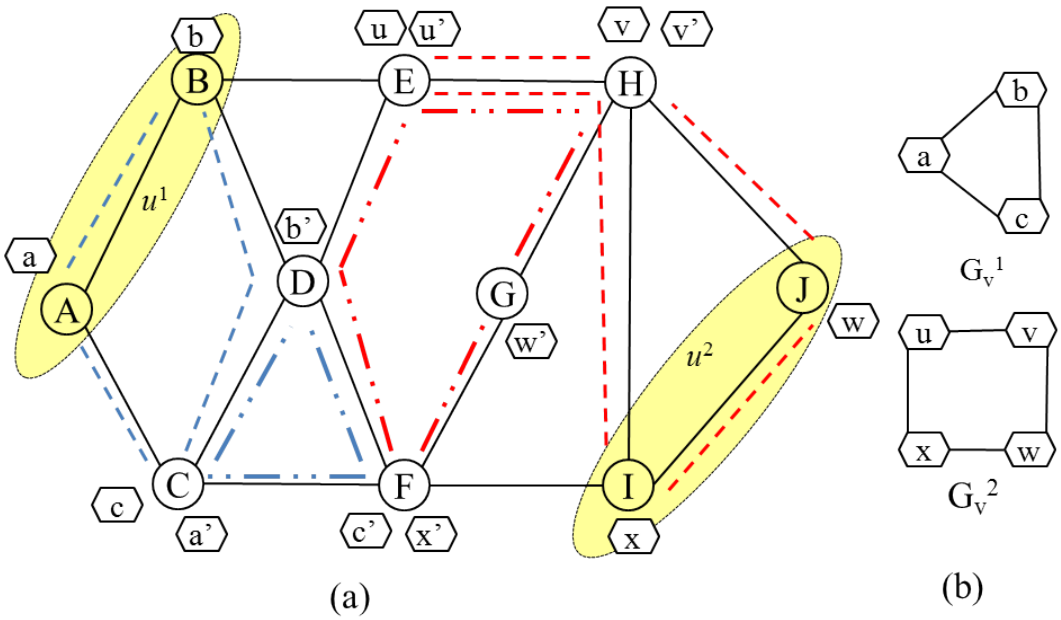



Figure 3.4: (a) substrate w. 2 failure regions, (b) 2 VN requests

node and its attached VN links and has been shown to outperform the well-studied R-ViNE algorithm in [NC01]. The NSVIM algorithm is also detailed further in the Appendix, and the region disjoint solution is now presented.

### 3.3.1 Failure Region Group Based Mapping (FRGBM)

The overall *failure region group based mapping* (FRGBM) scheme is shown in Figure 3.5. The algorithm starts by first separating the failure regions, $U$, into two static disjoint sub-groups, $\mathbb{G}^1$ and $\mathbb{G}^2$. The goal here is to cluster topologically-closer regions together and simplify subsequent backup VN mapping procedures. These sub-groups are then used to compute two failure region-disjoint VN mappings, $Z_1$ and $Z_2$, and resource sharing is also applied to reduce overall usages. Further details on each of these key steps are now presented.

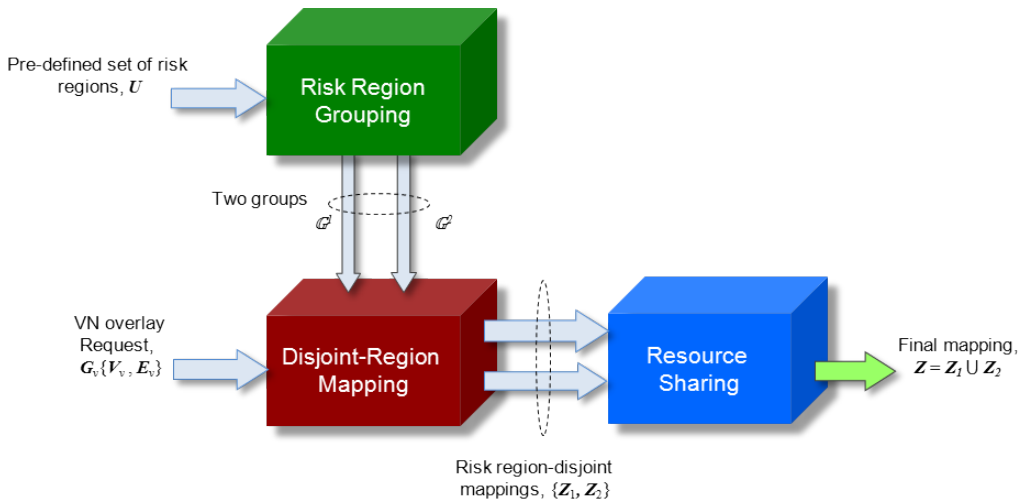



Figure 3.5: Overview of failure region group based mapping (FRGBM) scheme

The risk region grouping algorithm (Figure 3.5) uses a simple heuristic strategy based upon average distances. Namely, a distance matrix, $D$, is first computed to obtain the average distances between each failure region, i.e., $D = \{d_{ij}\}$ where $d_{ij}$ is the distance between regions $u^i$ and $u^j$. This step is shown in Figure 3.6, where each $d_{ij}$ value is computed by looping over nodes in regions $u^i$ and $u^j$. After this matrix has been computed, the next step separates the failure regions into two groups, $\mathbb{G}^1$ and $\mathbb{G}^2$, as shown in Figure 3.7. Initially, the two failure regions with the largest distance are determined and used to seed the $\mathbb{G}^1$ and $\mathbb{G}^2$ groups, respectively. The distances of all other "non-grouped" failure regions to $\mathbb{G}^1$ and $\mathbb{G}^2$ are then computed by averaging distances to existing grouped failure regions in the respective groups. These regions are then placed in the group with the shorter average distance. An example of this grouping algorithm is shown in Figure 3.8 for a 10-node network with 3 failure regions. In particular, the maximum $d_{ij}$ value here is 3.5, i.e., distance between region 1 and 3. Hence these regions are used to seed the initial groupings, i.e., $u^1 \in \mathbb{G}^1$ and $u^3 \in \mathbb{G}^2$. Carefully note that the failure region grouping algorithm can itself be defined as an ILP optimization problem as well. However, this is not

```
1: for i=0 to |U|
2:   for j=0 to |U|
3:     dist = 0
4:     for each v_s^i ∈ u^i
5:       dist_i = 0
6:       for each v_s^j ∈ u^j
7:         dist_ij=shortest path length from v_s^i to v_s^j
8:         dist_i=dist_i+dist_ij
9:       dist_i=dist_i / (number of nodes in u^j)
10:      dist=dist+dist_i
11:    d_ij=dist / (number of nodes in u^i)
```

Figure 3.6: Algorithm for computing average distance matrix, $D$

considered further for simplicity's sake.

Overall, the proposed FRGBM scheme uses the above failure region groupings to

```
1: G^1 = ∅, G^2 = ∅
2: Find maximum d_ij in D
3: Add u^i into G^1 and u^j into G^2
4: for i=0 to |U|
5:   if u^i ∉ G^1 and u^i ∉ G^2
6:     dist1=0, dist2=0, num1=0, num2=0
7:     for each u^j ∈ G^1
8:       dist1=dist1+d_ij
9:       num1++
10:    dist1=dist1 / num1
11:    for each u^j ∈ G^2
12:      dist2=dist2+d_ij
13:      num2++
14:    dist2=dist2 / num2
15:    if dist1<dist2
16:      add u^i to G^1
17:    else
18:      add u^i to G^2
```

Figure 3.7: Failure region grouping algorithm

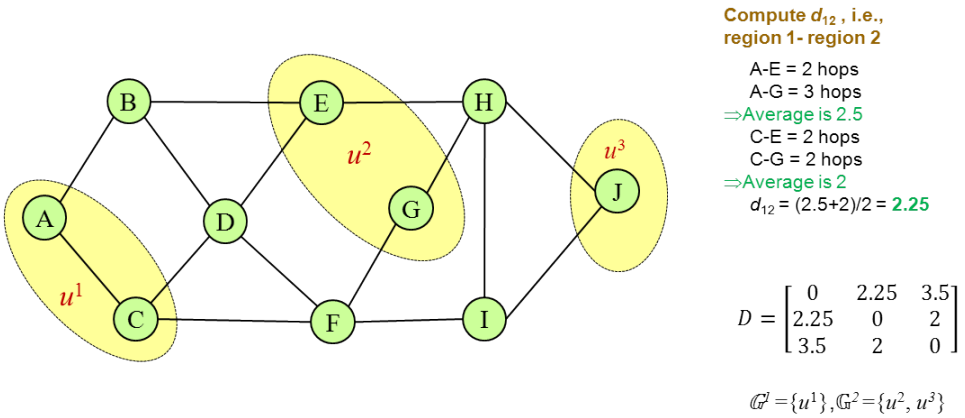



Figure 3.8: An example of computing failure region groups

compute two separate working/protection VN mappings, Figure 3.5. This algorithm is detailed in Figure 3.9 and starts out by pruning all failure regions in $\mathbb{G}^1$. A variation of the NSVIM algorithm, termed as *connectivity-aware NSVIM* (C-NSVIM), is then run to find a working VN mapping, $Z_1$. The $\mathbb{G}^1$ regions are then restored and those in $\mathbb{G}^2$ are pruned and the C-NSVIM algorithm re-run to compute the protection VN mapping, $Z_2$. Now the key difference between the NSVIM and modified C-NSVIM algorithm is an added constraint for computing the candidate substrate node list for each VN node, $\mathbb{L}$. Namely, the maximum connectivity of a candidate node is also checked, in addition to its node resources and adjacent link bandwidth constraints, i.e., to avoid mapping VN nodes to areas isolated by pruned failure regions. For example, consider the network in Figure 3.2 again. Here if failure regions $u^1$ and $u^2$ belong to the same group ($\mathbb{G}^1$ or $\mathbb{G}^2$) and are pruned during computation, then substrate nodes $v_s^0$ and $v_s^1$ will no longer be feasible for mapping VN requests with three or more nodes, i.e., since $v_s^0$ and $v_s^1$ will be isolated after pruning. Hence if the maximum connectivity number, $\tau$, of a substrate node is less than the number of VN nodes in the request, this node is not considered as a valid mapping candidate. Note that the connectivity of each substrate node after pruning $\mathbb{G}^1$ or $\mathbb{G}^2$ is fixed since these two sets are pre-computed at startup. Complete details on the NSVIM

and C-NSVIM schemes are also presented in the Appendix.

Finally, the FRGBM scheme also implements node and link bandwidth resource sharing between the working and protection VN mappings. This is possible since only one failure event is assumed to occur at a given time, a realistic assumption as catastrophic events are relatively rare. Now in order to implement this feature, an added data structure is introduced to record the substrate node and link resource allocations for each mapping, i.e., $Z_i$ in Step 6, Figure 3.9. Resource sharing is then done by selecting the maximum resource usage in the respective substrate node (link) amongst the two mappings, i.e., Step 9, Figure 3.9.

Now consider the overall computational complexity of the FRGBM scheme. Foremost, the grouping algorithm in Figure 3.7 has a complexity of $O(|U|^2)$, i.e., as it loops over all failure regions. Similarly, the distance matrix computation algorithm in Figure 3.6 also has a computation complexity of $O(|V_s|^2|E_s|log|V_s|)$. Finally the VN mapping procedure uses a modified C-NSVIM algorithm with a complexity of $O(|E_v|(2+|V_s|)|V_s||E_s|log|V_s|)$. Now since the failure region grouping is static, the related algorithm only needs to be run once before start up. Therefore the *run-time* complexity of the FRGBM scheme is bounded by $O(|E_v|(2+|V_s|)|V_s||E_s|log|V_s|)$.

### 3.3.2 Dynamic Failure Region Disjoint Mapping (DFRDM)

As noted above, the FRGBM scheme uses fixed failure region groupings that are determined at startup. However since network substrate (node and link) loads can vary dynamically, these fixed groupings can also lead to increased resource inefficiency. Therefore, in order to address this concern, a modified *dynamic failure region disjoint mapping* (DFRDM) scheme is proposed here, as shown in Figure 3.10.

The main idea behind the dynamic approach is to first compute a working VN mapping and then prune all the failure regions used by this mapping. Subsequently,

1: Define temporary substrate graph $G_t(V_t, E_t)$ with $V_t$ and $E_t$
2: **for** $i$=1 to 2
3: Copy current node/link resource from $G_s$ to $G_t$
4: Prune each failure region $u^j \in \mathbb{G}^i$ in $G_t$
5: Run C-NSVIM on $G_t$ for mapping $Z_i$, if failed return FAIL
6: Record substrate node/link resource allocation for $Z_i$
7: Restore each failure region $u^j \in \mathbb{G}^i$ in $G_t$
8: Assign node/link resource sharing between $Z_1$ and $Z_2$ according to records generated in Step 6 and compute final resource usage
9: Reserve node/link resource in $G_s$ according to final resource usage computed in Step 9
10: Return SUCCESS

Figure 3.9: Failure region group based mapping (FRGBM) algorithm

the protection mapping is computed. However, since the working VN mapping may span a large number of failure regions, this approach may restrict the available pool of resources for the protection mapping. As a result, additional penalty costs are introduced to prevent the working VN mapping from spanning too many failure regions. Specifically, the NSVIM algorithm is still re-used here, but an additional

1: Define temporary substrate graph $G_t(V_t, E_t)$ with $V_t$ and $E_t$
2: Copy current node/link resource from $G_s$ to $G_t$
3: Run NSVIM with penalty cost on $G_t$, if failed return FAIL
4: Record substrate node/link resource allocation for $Z_1$
5: Copy current node/link resource from $G_s$ to $G_t$
6: Prune failure regions accessed by $Z_1$ in $G_t$
7: Run C-NSVIM on $G_t$ for mapping $Z_2$, if failed return FAIL
8: Record substrate node/link resource allocation for $Z_2$
9: Restore failure regions accessed by $Z_1$ in $G_t$
10: Assign node/link resource sharing between $Z_1$ and $Z_2$ according to records generated in Steps 4 and 8, compute final usage
11: Reserve node/link resource in $G_s$ according to final resource usage computed in Step 10
12: Return SUCCESS

Figure 3.10: Dynamic failure region disjoint mapping (DFRDM) algorithm

penalty cost, $P(v_s)$, is added if a candidate node $v_s$ is located in a new/different failure region from the regions already covered by the mapping. Another penalty cost, $P(e_s)$, is also added if the path mapping for this VN node (to a mapped adjacent VN node) traverses any new failure regions. Note that an added data structure is also needed here to record all the failure regions accessed by a VN request, i.e., Step 3, Figure 3.10.

Once the working VN mapping has been computed, the protection mapping is determined by running the C-NSVIM algorithm, but without penalties. Akin to the FRGBM scheme, node connectivity is also used here to avoid isolating candidate nodes with low connectivity. Namely, a static connectivity status is computed for each substrate node for all possible failure region pruning scenarios in the initial stage. In particular, let the number 1 (0) denote a pruned (non-pruned) failure region. Using this representation, a failure region pruning scenario can be expressed as a $|U|$-dimensional vector binary $< a_{u^1}, a_{u^2}, ..., a_{u^{|U|}} >$ with the total number of potential scenarios given by $2^{|U|}$. Now for each scenario, the number of substrate nodes that can be reached after pruning the corresponding failure regions, i.e., maximum connectivity number $\tau$, can be computed for every substrate node by using a breadth-first search. Hence by searching the pre-computed connectivity status values, substrate nodes with $\tau$ values smaller than the number of VN nodes in a VN request will not be considered as candidates. Carefully note that connectivity information is only computed once at initialization and hence this will not affect run-time complexity, i.e., akin to the FRGBM scheme. Hence the DFRDM scheme also has the same overall complexity as the FRGBM scheme, i.e., $O(|E_v|(2+|V_s|)|V_s||E_s|log|V_s|)$.

### 3.3.3 Load Balancing

Overall, both the FRGBM and DFRDM algorithms compute substrate link and node costs according to pure *"mapping cost"* (MC) values. However, since these values are usually static (operator-specified), they cannot account for real-time node and link bandwidth resource loads at the substrate level. In turn this may yield increased congestion at specific nodes or links. Hence a simple *load balancing* (LB) strategy is proposed here to alleviate such concerns. Namely, substrate link costs (weights) are defined as inversely-proportional to the load as follows:

$$C(e_s) = \frac{B_c}{B(e_s) + \sigma} \qquad \text{(Eq. 3-23)}$$

where $B_c$ is the full capacity of a substrate link and $\sigma$ is a small value (to avoid division errors). Similarly, node resources costs can also be defined as:

$$C(v_s) = \frac{R_c}{R(v_s) + \sigma} \qquad \text{(Eq. 3-24)}$$

where $R_c$ is the full resource capacity of a substrate node.

## 3.4 Performance Evaluation

The performance of the proposed survivable VN mapping schemes is now tested using customized $OPNETModeler^{TM}$ models. Namely, two substrate topologies are tested here, including the smaller 10-node network with 3 failure regions in Figure 3.1a (with modified nodal resource/link capacities) and the larger 24-node network with 5 failure regions in Figure 3.2. Here, all substrate nodes have 100 units of resource capacity and all substrate links have 10,000 units of bandwidth. Meanwhile the VN requests are varied between 3-5 nodes and 4-7 nodes each for the 10- and 24-node topologies, respectively. Average VN topology node degrees are also set to 2.3 and 2.6 for these two networks. Meanwhile, requested VN node capacities

are uniformly distributed between 1-10 units, and requested VN link capacities are uniformly distributed between 50-1,000 units. All requests follow random exponential holding and inter-arrival times, with means $\mu$ and $\lambda$, respectively. In particular, a value of $\mu = 600$ time units is chosen here, and $\lambda$ is adjusted according to load. Meanwhile, the load is measured using a modified Erlang metric by accounting for the VN request size, i.e., by taking the product of average number of VN links and $\mu/\lambda$ . Finally, the *CPLEX 12.4* tool is incorporated with $OPNETModeler^{TM}$ in order to solve the MILP formulation, i.e., per incoming VN request.

For comparison purposes, the SOUM and IOCM survivability schemes in [HY04] are also tested here. Furthermore, all schemes are gauged using both the MC and LB cost assignment strategies. Now tests for the 10-node topology are done using 10,000 random VN requests, whereas tests for the 24-node network are done using 100,000 random requests (heuristic schemes only). By contrast, the MILP optimization is only tested for the 10-node network owing to excessive computational complexity. Failure events are also randomly triggered after an average of 1,000 incoming VN requests, i.e., a failure region is selected in random uniform manner and all of its nodes and links are failed.

### 3.4.1 Blocking Rates

The overall request blocking rates are shown in Figure 3.11 for the various schemes. First of all, the findings for the 10-node topology in Figure 3.11a show that the MILP optimization scheme gives the lowest blocking of all. In fact, this solution also yields the best results for nearly all of the other metrics evaluated. In general, these behaviors are expected since the MILP is an optimal strategy, and hence the remaining discussions only focus on the heuristic strategies.

Now for the smaller 10-node network, results indicate very little separation be-

tween the MC-based heuristic schemes, i.e., with IOCM-MC and FRGBM-MC giving slightly lower blocking than the others. However, for the larger 24-node network, Figure 3.11b, the proposed FRGBM and DFRDM schemes give much lower blocking than their IOCM and SOUM-based counterparts, e.g., about 94% lower blocking with FRGBM-MC versus SOUM-MC at low-medium loads. These gains are due to the fact that the 24-node topology has much more substrate nodes and links, and this allows the failure region-disjoint schemes to utilize network resources in a more efficent manner. This contrasts with the SOUM and IOCM schemes which exhaustively provision protection for all possible failure regions. In addition, the FRGBM-MC scheme actually gives lower blocking than the dynamic DFRDM-MC except at heavy loads. The reason here is that the fixed cost computation strategies do not use any real-time substrate load information (and hence cannot efficiently generate two disjoint failure regions). Finally, the results show that the LB-based heuristics yield the lowest blocking of all heuristics. These findings confirm that distributing VN loads across network substrate nodes/links can generally improve blocking performance. Furthermore, the DFRDM-LB scheme also gives lower blocking than FRGBM-LB, especially for the 24-node topology, i.e., as it performs dynamic mapping to disjoint failure regions (and can further leverage real-time substrate load information).

### 3.4.2 Long Term Revenue

The long term revenues are also plotted in Figure 3.12 and show that the FRGBM and DFRDM schemes achieve much higher values than the competing SOUM and IOCM solutions, i.e., for both the MC and LB variants. For example the FRGBM-MC scheme gives almost 44% more revenue than SOUM-MC in the 24-node topology at higher loads, see Figure 3.12b. In addition, the LB-based variants also provide a significant increase in revenues as compared to the fixed weight MC-based schemes. Carefully note that revenue discrepancies also decrease with load since request block-

ing rates are lower.

### 3.4.3 Long Term Cost

Long term costs are further plotted in Figure 3.13 for both networks. In comparison with the SOUM and IOCM strategies, here the FRGBM and DFRDM heuristic schemes show slightly higher costs with the MC weighting approachs. Also, all LB-based heuristics yield higher costs than their MC counterparts. This is expected as these former schemes accept more VN requests (lower blocking), which in turn leads to additional resource consumption and increased costs. In light of this, long term cost may not a proper metric for evaluating VN mappings, i.e., as lower costs can possibly result from lower acceptance ratios. Hence the average cost per VN is also evaluated, as detailed next.

### 3.4.4 Average Cost

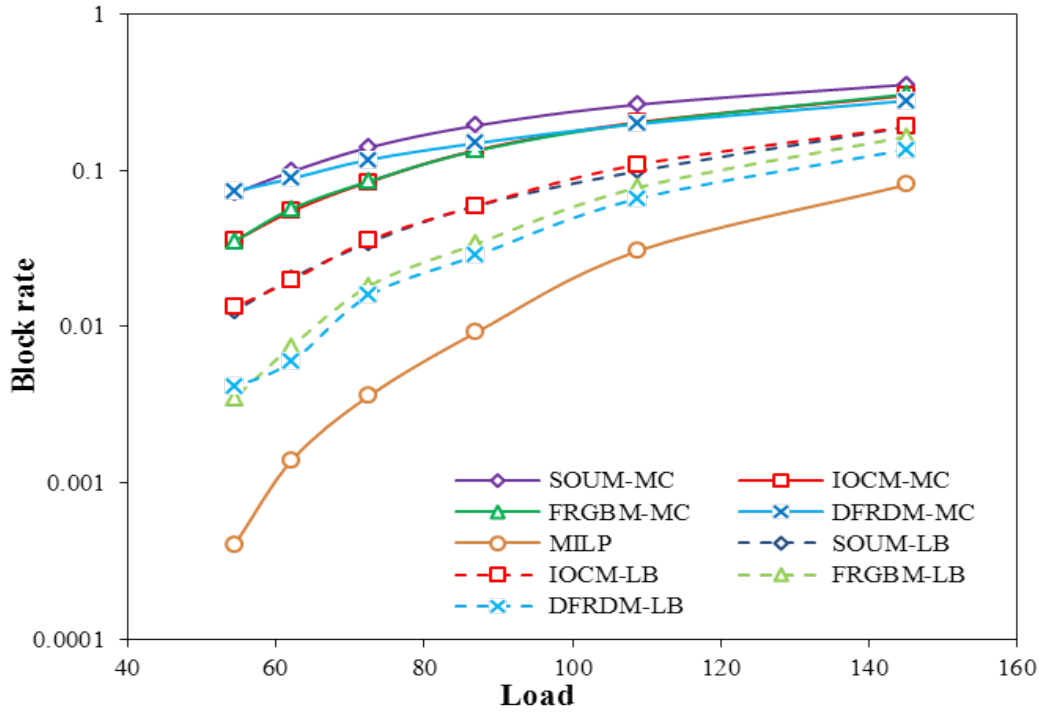
The average costs per VN are plotted in Figure 3.14. These results show that both the FRGBM and DFRDM schemes can achieve lower average cost than SOUM and IOCM (except with MC-based heuristics in the 10-node topology). Average costs also decline under heavier loads. This reduction is due to the fact that VN requests with larger numbers of VN nodes/links are more likely to be accepted under lighter loads. Hence these types of requests will consume more resources, leading to higher average per VN cost. To show this more clearly, the average revenue per VN is plotted in Figure 3.15, and the results indicate that revenue is high when traffic is light and low when traffic is heavy. Now since the revenue for each VN is only related to the VN topology, this shows that larger VN requests are accepted more easily when traffic is low. Finally, these findings also indicate that the FRGBM and DFRDM schemes can support larger VN sizes than the SOUM and IOCM schemes.

### 3.4.5 Net Revenue

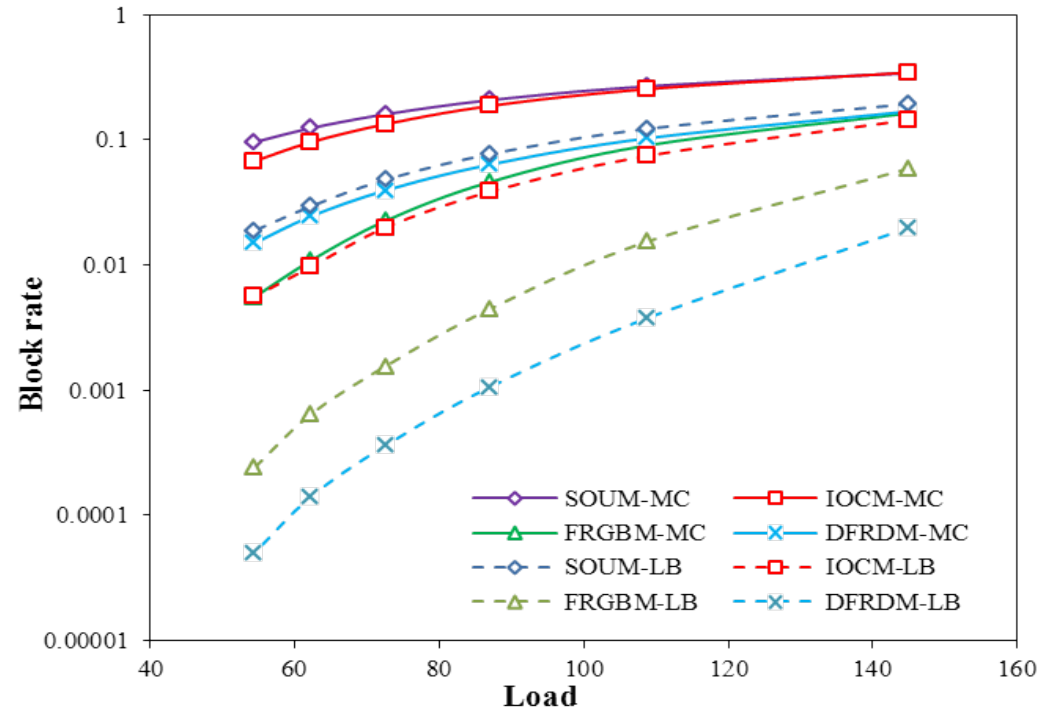
The net revenues from Eq. 3-5 are also plotted in Figure 3.16 and clearly show that the proposed FRGBM and DFRDM schemes generate significantly-higher revenues than SOUM and IOCM (for both the MC- and LB-based variants). These results also validate the gains of using a failure region-disjoint mapping approach. In addition, the LB-based heuristics tend to give higher net revenues as well (versus their MC-based counterparts). The only exception here is the SOUM-LB scheme, which gives rather low revenues. The main reason here is that LB-based strategies yield longer path routes for the mapped VN link connections. Compounding this increase is the fact the SOUM scheme exhaustively computes multiple protection mappings for all failure regions (and combines them via sharing). As noted earlier, net revenues also decrease with traffic load, i.e., akin to Figure 3.12.

### 3.4.6 VN Failures and Node Migrations

The number of failed VN requests is also plotted in Figure 3.17 to measure overall robustness. These results show that the proposed FRGBM scheme is the most resilient of all, as it groups failure regions and prevents the working mapping from spanning across too many regions. For example, the FRGBM-MC scheme gives at least 55% fewer VN requests failures versus the SOUM and IOCM schemes in the 24-node network, Figure 3.17b. In addition, the total number of post-fault VN node migrations are also plotted in Figure 3.18 in order to gauge operational complexity. Again, the FRGBM (and IOCM) schemes give the lowest number of migrations since they experience the lowest number of VN failures. For example, the FRGBM-LB scheme gives 44% fewer migrations than the DFRDM-LB scheme at high loads in the 24-node topology, see Figure 3.18b.

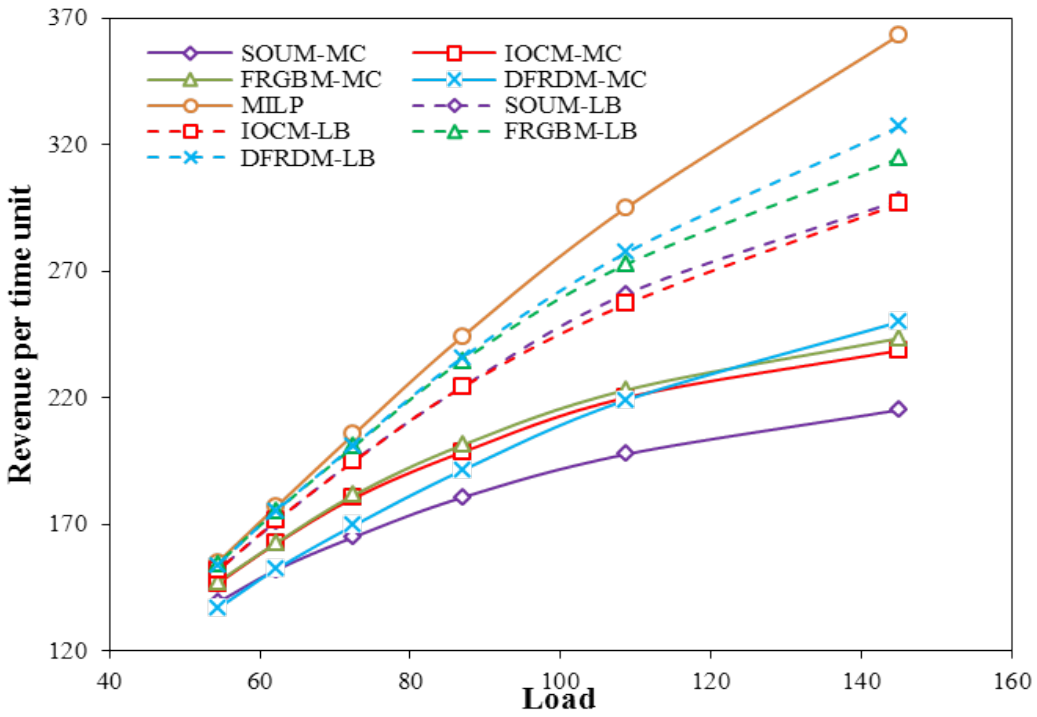



(a)

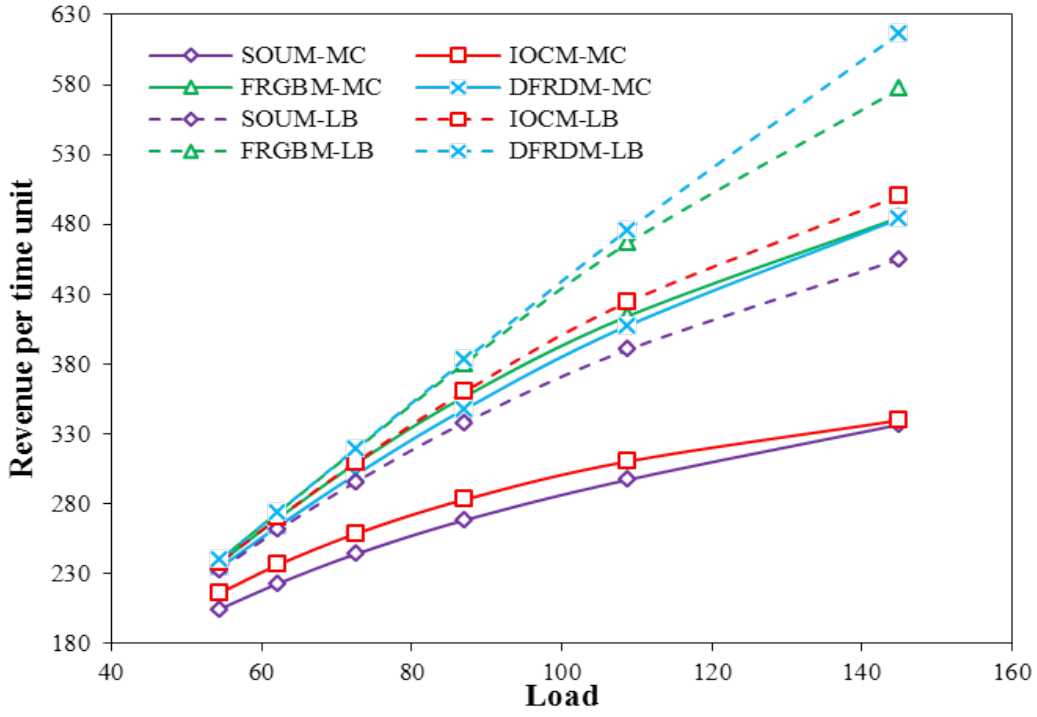



(b)

Figure 3.11: Blocking rate: a) 10-node topology, b) 24-node topology

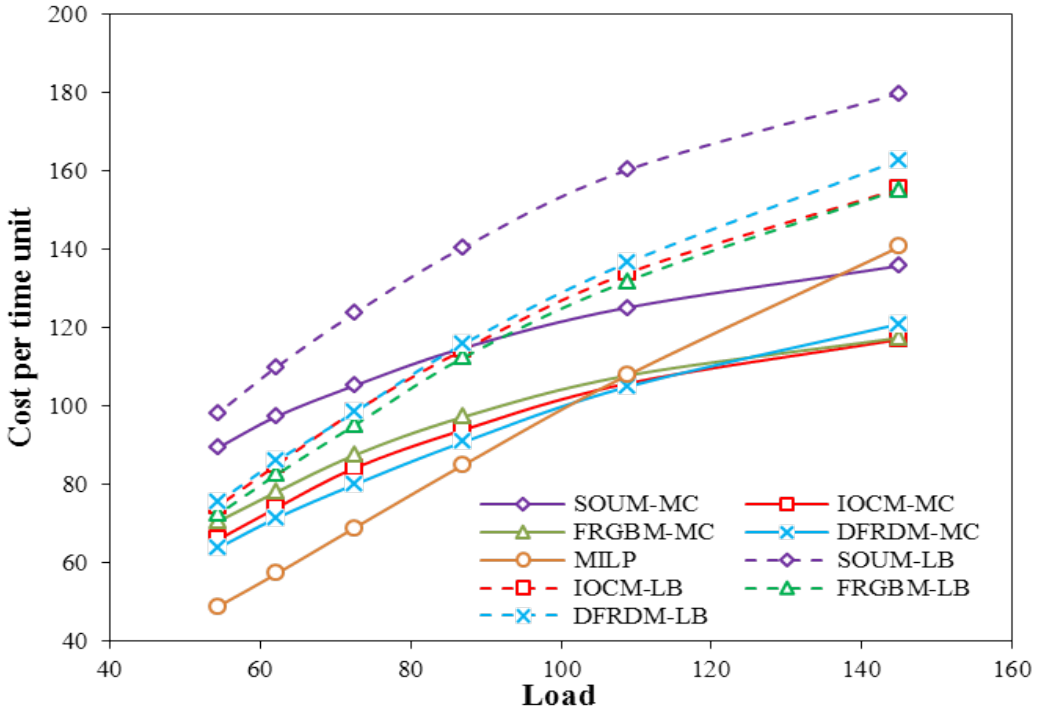



(a)

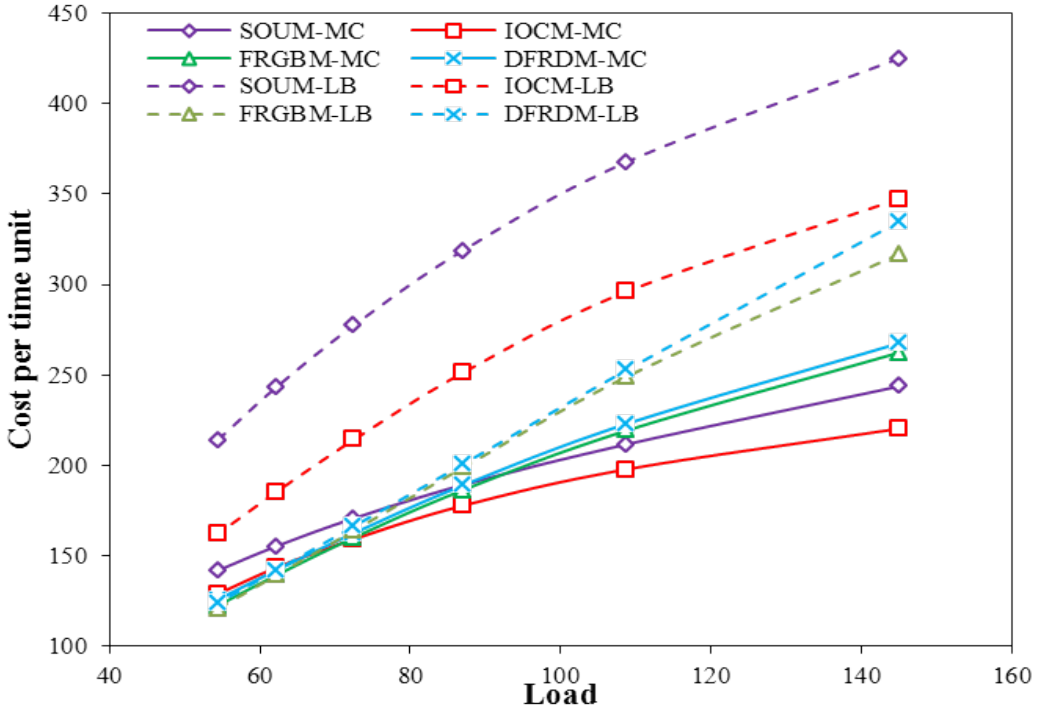



(b)

Figure 3.12: Long term revenue: a) 10-node topology, b) 24-node topology

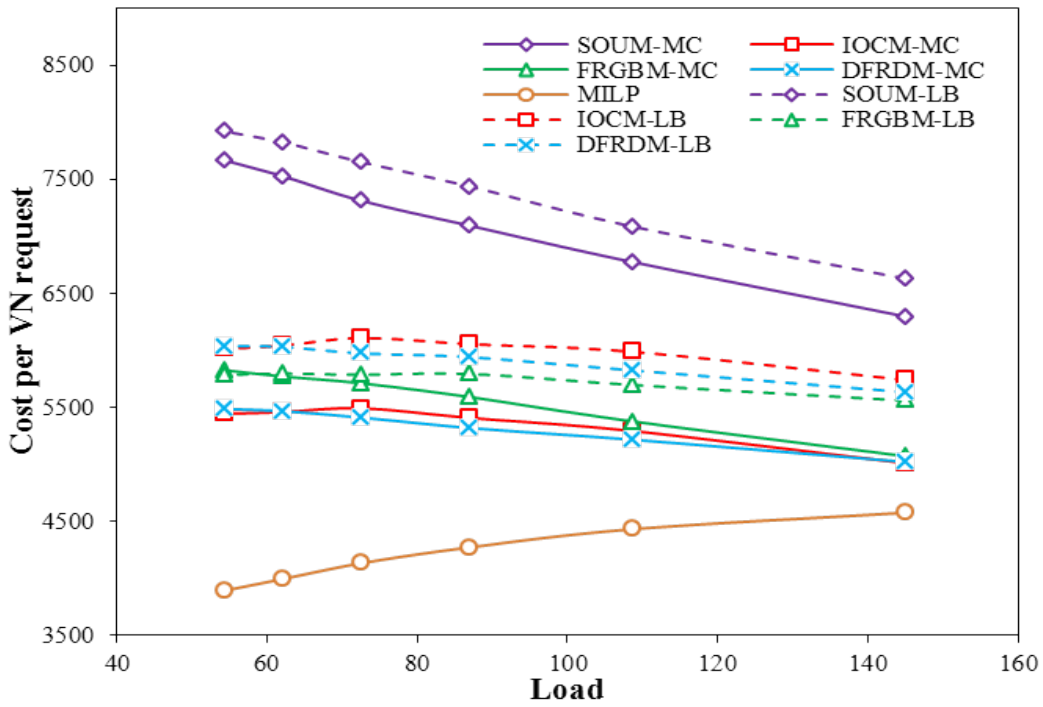



(a)

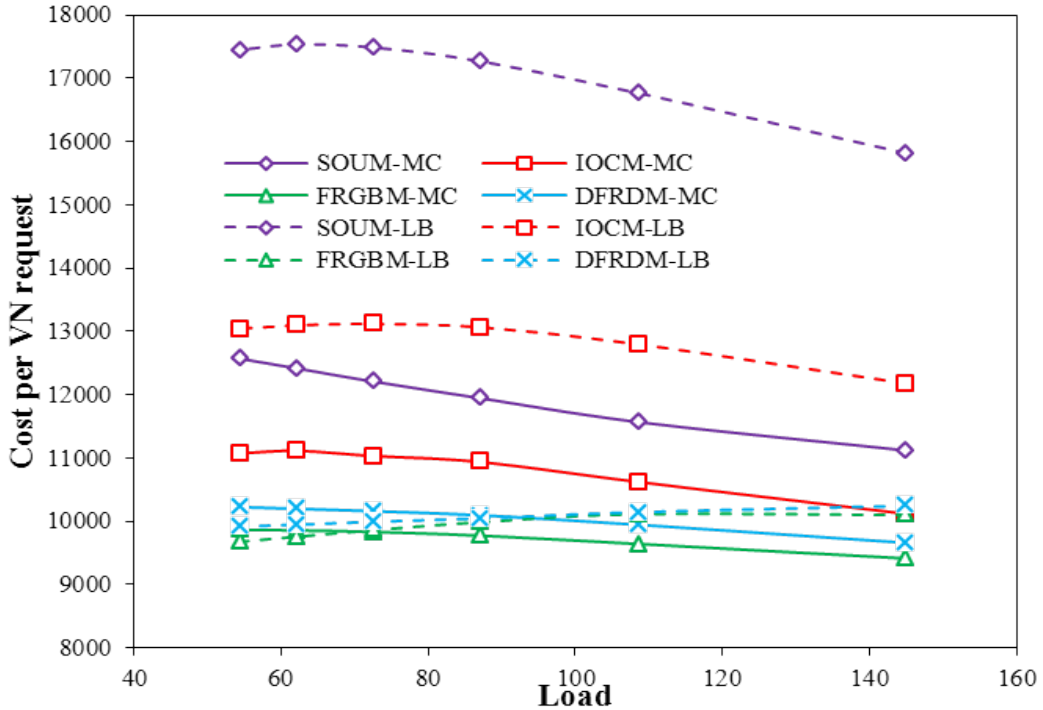



(b)

Figure 3.13: Long term cost: a) 10-node topology, b) 24-node topology

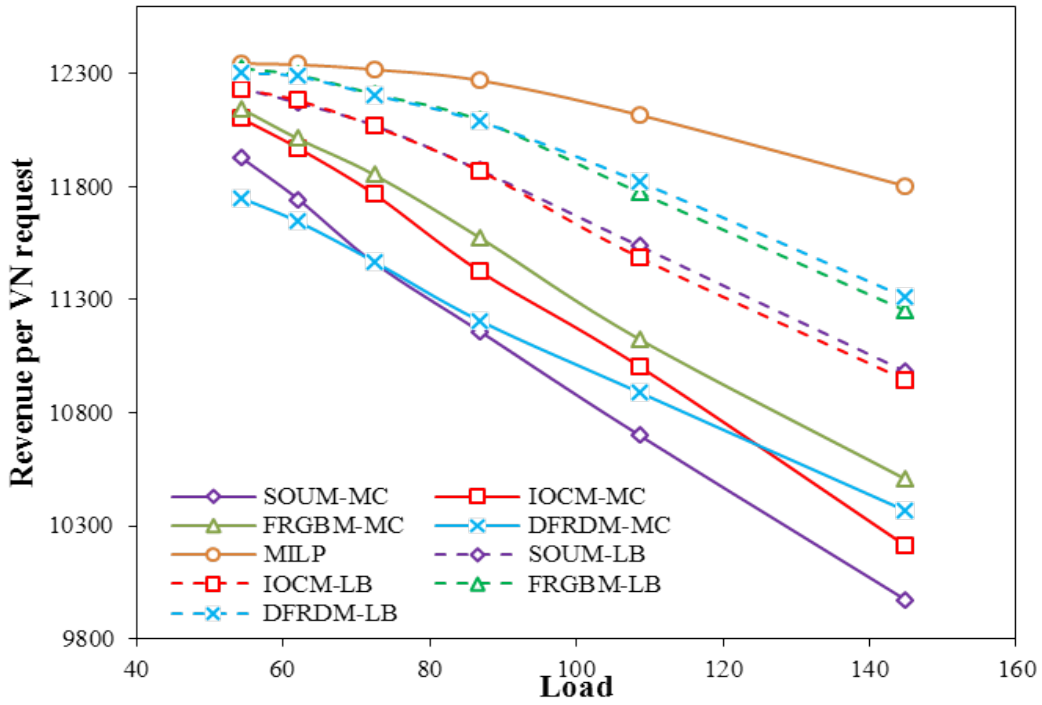



(a)

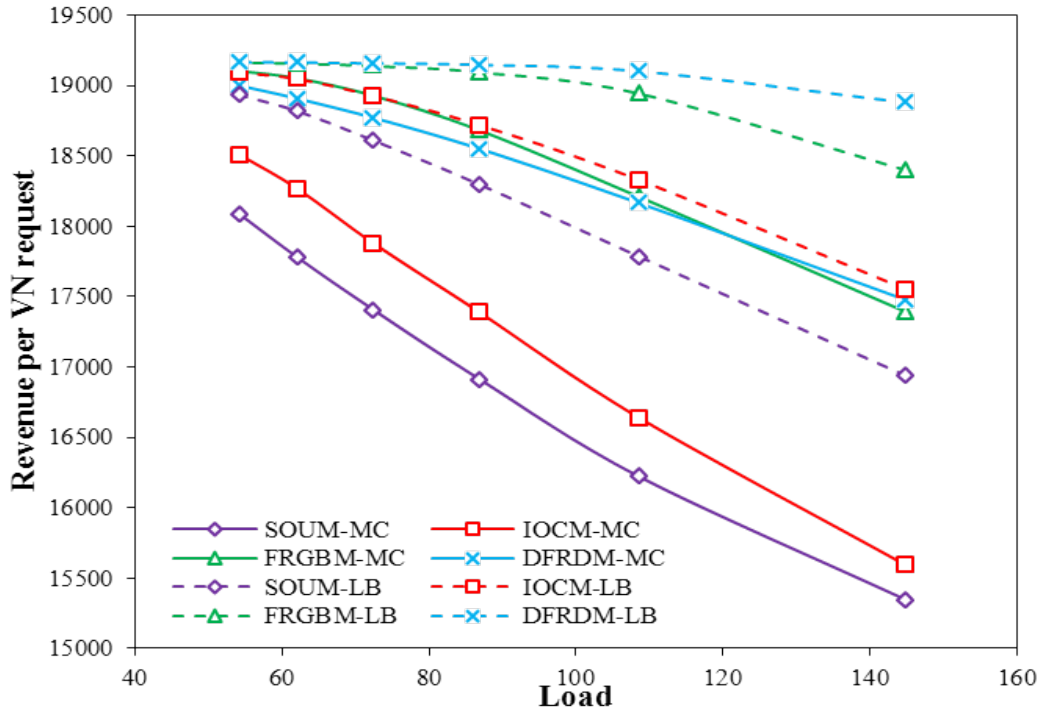



(b)

Figure 3.14: Cost per VN: a) 10-node topology, b) 24-node topology

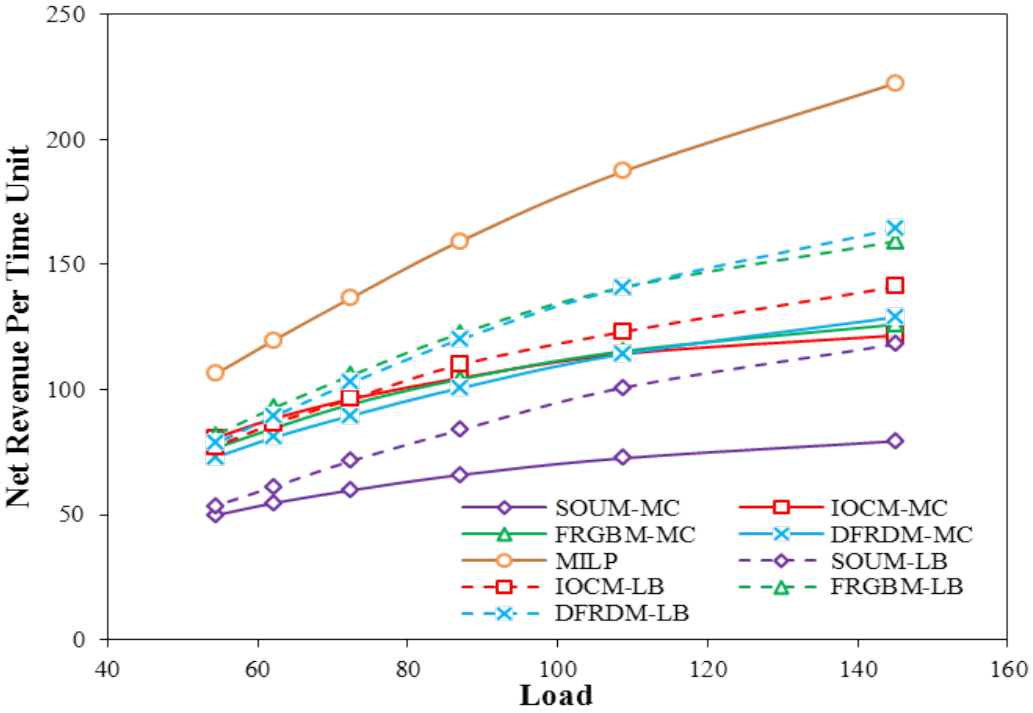



(a)

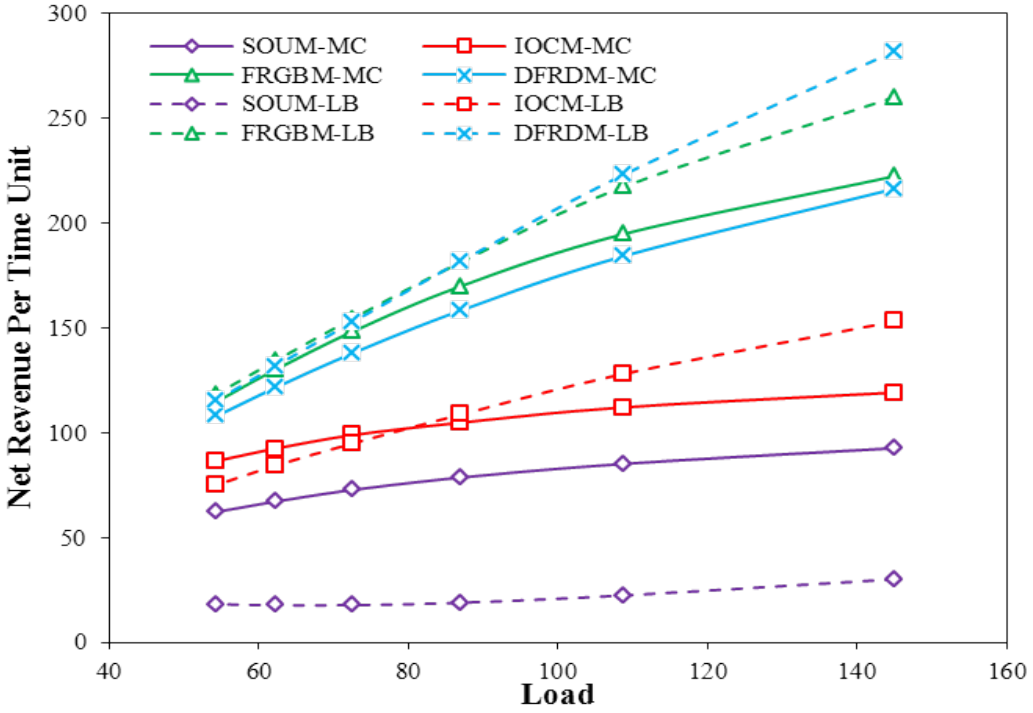



(b)

Figure 3.15: Revenue per VN: a) 10-node topology, b) 24-node topology

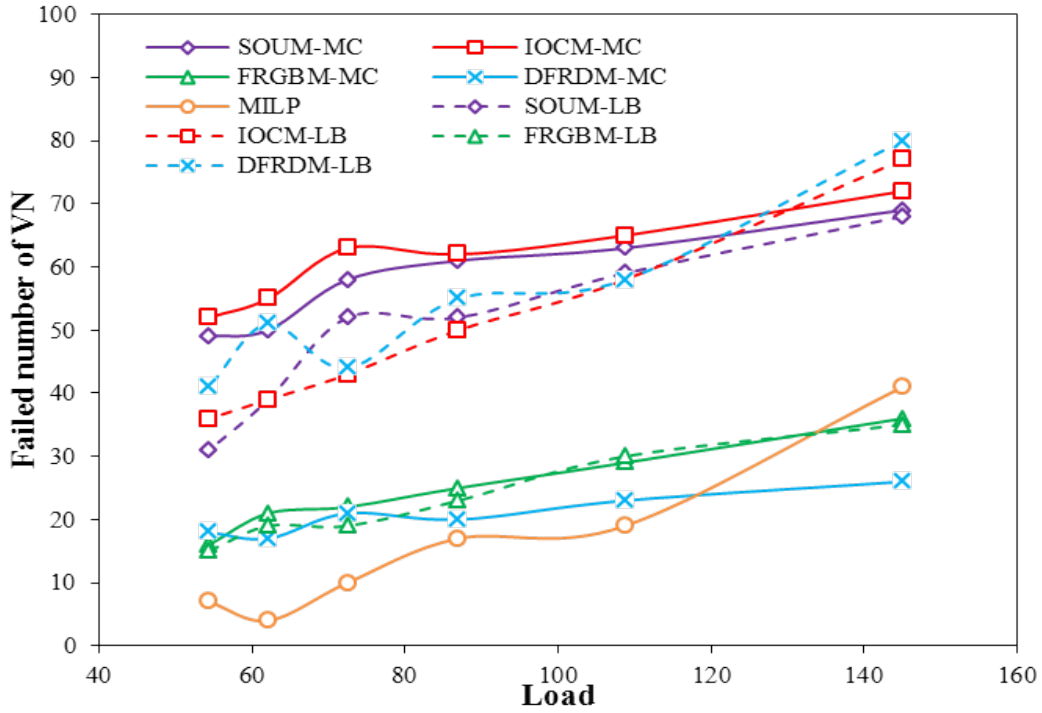



(a)

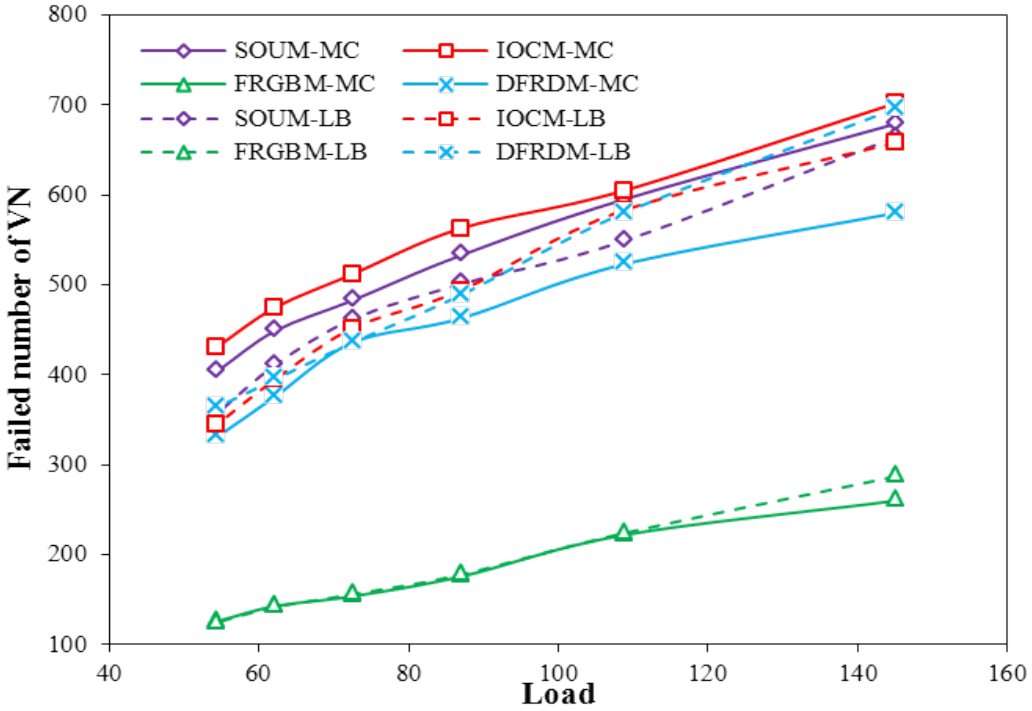



(b)

Figure 3.16: Net revenue: a) 10-node topology, b) 24-node topology

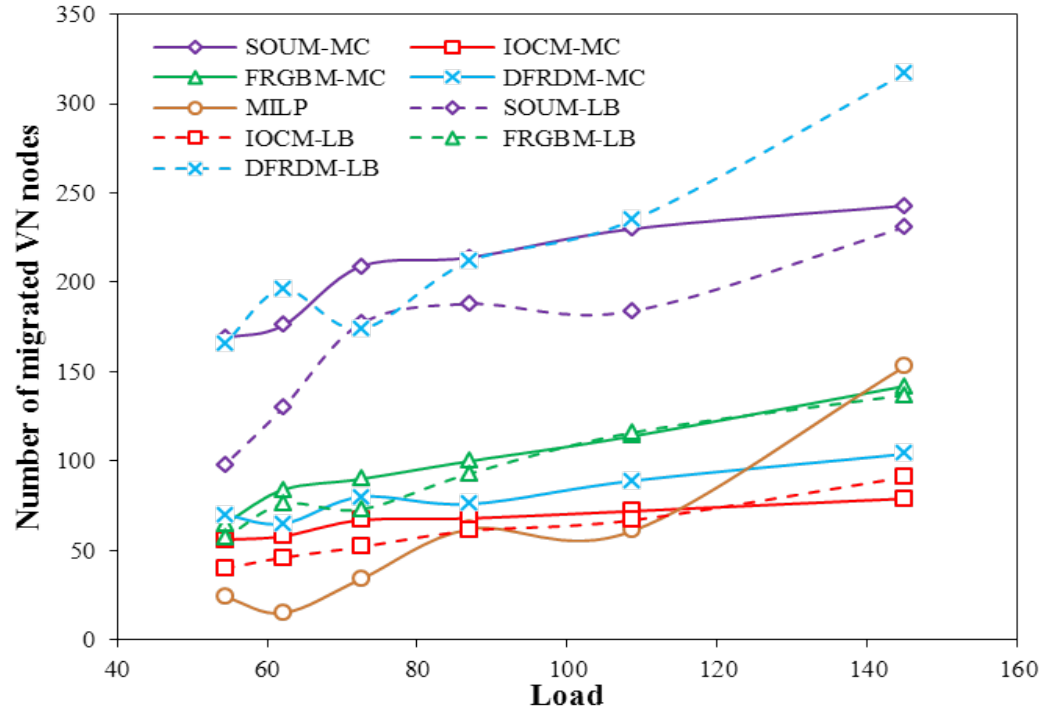



(a)

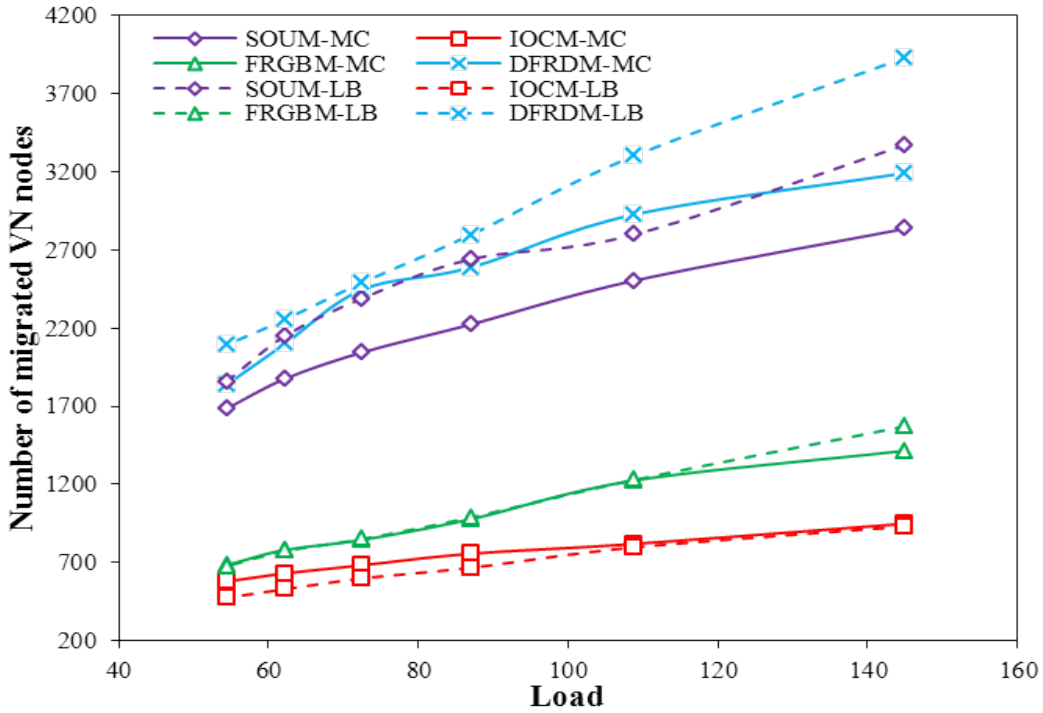



(b)

Figure 3.17: Failed number of VN: a) 10-node topology, b) 24-node topology




(a)




(b)

Figure 3.18: Num. of migrated VN nodes: a) 10-node topology, b) 24-node topology

# Chapter 4

# VN Provisioning for Probabilistic Failures

In general, pre-provisioned VN protection schemes consume higher amounts of resources since backup node and link capacities must be reserved in advance. This may lead to higher resource inefficiency and reduced operator revenues, especially when the relative frequency of failure events is low, i.e., particularly true for catastrophic stressors. Moreover, many mid-tier cloud users may prefer more economically-priced service offerings with reduced levels of stringency, i.e., able to tolerate infrequent service outages. Hence for these service types, provisioning full service protection may not be a very feasible/economic option. Instead, it may be much more beneficial to provision basic working mode services with increased awareness (resiliency) to rare catastrophic events. However, this approach necessitates a *probabilistic* treatment of failure events, a topic which has not been studied within the context of VN provisioning, i.e., VN risk minimization. Consider some further details here.

Overall, the key objective in VN risk minimization is to perform VN node/link mappings so as to minimize the risk of VN failure under a set of pre-defined stres-

sors. Now since a VN topology is comprised of both VN nodes and VN links, the risk associated with its mapping will be related to the underlying failure probabilities of the substrate nodes and links that it uses. Additionally, VN links will be susceptible to both substrate link and node failures. Along these lines, this chapter introduces probabilistic substrate node/link failure information into the VN mapping process. Using this information, two "risk-aware" VN mapping heuristic schemes are presented. Namely, the first heuristic strictly focuses on minimizing the risk of a working VN mapping. However, since pure risk minimization objectives have been known to yield very high resource inefficiency in regular connection routing scenarios [OD01], a second solution is also proposed to jointly incorporate both risk minimization and *traffic engineering* (TE) efficiency concerns. Carefully note that optimization-based schemes are not considered here since the problem is non-linear and difficult to solve due to high complexity. The heuristics solutions are now presented.

## 4.1 Network Model and Description

The overall notation and network model for VN risk minimization is similar to that introduced in Section 3.1. In addition, many of the performance evaluation metrics defined for VN protection are also applicable here. However, in order to properly describe the characteristics of stressors, further failure probability information needs to be introduced. Consider the details here.

Akin to the stressor model in Section 3.1, a finite number of failure regions are enumerated in a set $U$. Now owing to the probabilistic nature of these events, each stressor $u \in U$ has an occurrence probability given by $w_u \in [0, 1]$, which in turn yields a *failure region probability vector* for all events, i.e., $\vec{\boldsymbol{w}} = [w_{u^1}, w_{u^2}, ..., w_{u^{|U|}}]$. Furthermore, it is also assumed that large-scale stressor events are sufficiently rare so as to be mutually-exclusive, i.e., only one can occur at a given time ($\sum_{u \in U} w_u =$

1). Next, a conditional *risk vector* is also defined for each substrate node and link in the failure stressor graph $G_u(V_u, E_u)$, i.e., $\vec{\boldsymbol{p}}_{v_s} = [p_{v_s}^{u^1}, p_{v_s}^{u^2}, ..., p_{v_s}^{u^{|U|}}]$ and $\vec{\boldsymbol{p}}_{e_s} = [p_{e_s}^{u^1}, p_{e_s}^{u^2}, ..., p_{e_s}^{u^{|U|}}]$. Namely, $p_{v_s}^{u^i}$ ($p_{e_s}^{u^i}$) is the conditional node (link) failure probability if stressor $u_i$ occurs. Here it is also assumed that each conditional (node, link) failure probability is independent.

Next, some new metric definitions are introduced to handle more specialized VN risk minimization needs. Namely, the penalty associated with service disruption for a given VN request, $G_v$, is defined as:

$$PE(G_v) = \sum_{e_v \in E_v} b(e_v) * \mathbb{PE}(e_v) + \psi \sum_{v_v \in V_v} r(v_v) * \mathbb{PE}(v_v) \qquad \text{(Eq. 4-1)}$$

where $\mathbb{PE}(e_v)$ is the unit link penalty, $\mathbb{PE}(v_v)$ is the unit node penalty, and $\psi$ is the fraction of node penalty. Using the above, the net revenue in Eq. 3-5 is also re-defined as:

$$\frac{\sum_i (REV(G_v^i) - COST(G_v^i)) - \sum_j PE(G_v^j)}{T}, \forall G_v^i \in \mathbb{A}, \forall G_v^j \in \mathbb{F} \qquad \text{(Eq. 4-2)}$$

where $REV(G_v)$ and $COST(G_v)$ are defined in Eq. 3-1 and Eq. 3-2, respectively, $G_v^i$ is the *i-th* VN request, $\mathbb{A}$ is the set of all accepted VN requests, $G_v^j$ is the *j-th* VN request, $\mathbb{F}$ is the set of all failed VN requests, and $T$ is the total run time. Here net revenue is basically adjusted by the service disruption penalty in Eq. 4-1.

## 4.2 VN Risk Minimization

Based upon the revised notation above, two novel "risk-aware" VN mapping schemes are now proposed. Overall, these algorithms use heuristic methodologies to try to map VN requests and minimize their vulnerability to multi-failure stressors.

### 4.2.1 VN Embedding with Risk Minimization Only

Probabilistic VN embedding is first considered purely from a risk minimization perspective, i.e., *risk minimization mapping* (RMM) scheme. Namely, the objective of this scheme is to minimize the failure probability of a VN mapping given a predefined set of failure regions, $U$. Hence this algorithm starts by first defining a risk value for each substrate node (link) as the dot product of its node risk vector $\vec{\boldsymbol{p}}_{v_s}$ (link risk vector $\vec{\boldsymbol{p}}_{e_s}$) and failure region probability vector $\vec{\boldsymbol{w}}$, i.e., $\xi_{v_s} = \vec{\boldsymbol{p}}_{v_s} \cdot \vec{\boldsymbol{w}}$ for a substrate node and $\xi_{e_s} = \vec{\boldsymbol{p}}_{e_s} \cdot \vec{\boldsymbol{w}}$ for a substrate link. Subsequently, modified node and link risk values are defined as follows:

$$r_{v_s} = log\frac{1}{1-\xi_{v_s}} \qquad \text{(Eq. 4-3)}$$

$$e_{v_s} = log\frac{1}{1-\xi_{e_s}} \qquad \text{(Eq. 4-4)}$$

In particular, these values are chosen as they can be applied in an *additive* manner to compute the total failure risk of an end-to-end path (see Appendix B for detailed proof on equivalency between using $r_{v_s}$ and $e_{v_s}$ in place of $\xi_{v_s}$ and $\xi_{e_s}$ for shortest-path computation). These values are then assigned as appropriate weights in the substrate graph and used in the subsequent shortest-path computation steps. In particular, the risk for a VN node is defined as the failure risk, $r_{v_s}$, of the underlying substrate node to which it is mapped. Meanwhile, the risk for a VN link is defined as the *sum* of failure risks, $r_{v_s}$ and $r_{e_s}$, of all the substrate nodes and links along its connection route.

Now carefully note that traditional shortest-path algorithms, e.g., such as Dijkstra's, also do not handle node weights. However, node weights can still be incorporated by transferring the original graph to a directed graph and then adding the node weight (risk probability) to the ingress links. Subsequently, shortest-path algorithms can be run on this modified graph. An example of this transfer is illustrated

in Figure 4.1, in which the original weight of link $AB$ is 0.6, the weight of node $A$ is 0.3, and the weight of node $B$ is 0.1. Hence in the modified directed graph, the weight of link $AB$ is now set to $0.6+0.1 = 0.7$ and that for link $BA$ to $0.6+0.3 = 0.9$.

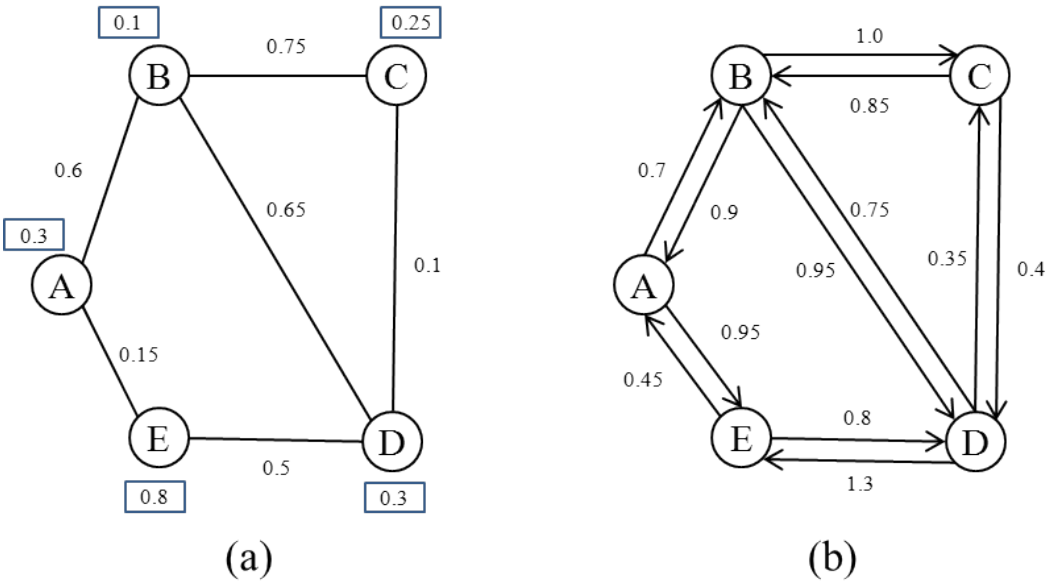



Figure 4.1: (a) A sample graph w. node weight (b) Transferred directed graph

The overall RMM scheme follows a joint VN mapping strategy and is shown in Figure 4.2. Namely, the algorithm basically sorts all VN nodes in descending order of node degree and then loops through to map each one, i.e., call to *FSN_RMM* routine, Figure 4.2. Now if a VN node is successfully mapped here, then all of its emanating VN links are also mapped (but only those whose other VN node end-points have already been mapped, see Steps 6-8 in Figure 4.2). After each iteration, the respective risk values for all assigned/traversed substrate nodes/links are also set to zero, i.e., since re-using them will not introduce any further risk. Now consider the details for the node mapping procedure.

The "risk-aware" node mapping scheme is now detailed, i.e., see Figure 4.3 for pseudocode of FSM_RMM subroutine. This algorithm first builds a candidate sub-

strate node list, $\mathbb{L}$, for each VN node, $v_v$, based upon three criteria, see Steps 2-9 in Figure 4.3. Foremost the available node resources at a candidate node $v_s \in V_s$ should not be less than the requested amount in $v_v$. Next, the maximum available bandwidth along all adjacent substrate links of $v_s$ must not be less than the maximum bandwidth requested by adjacent VN links of $v_v$. Finally, the total available bandwidth of an adjacent substrate link of $v_s$ should not be less than the total amount of bandwidth required along all adjacent VN links of $v_v$. Overall, these three criteria ensure a feasible mapping for VN node $v_v$.

Once the list $\mathbb{L}$ has been built, the VN node mapping algorithm loops through all candidate substrate nodes and selects one based upon three different cases. In the first case, if $v_v$ is the first VN node in the request then the cost values for all nodes $v_s \in \mathbb{L}$, $c_{v_s}$, are set equal to their risk values, $r_{v_s}$, see Step 12 in Figure 4.3. The substrate node mappings for $v_v$ is then done by randomly selecting a $v_s \in \mathbb{L}$ with selection probabilities set inversely-proportional to $c_{v_s}$, i.e., see Step 22 in Figure 4.3. This approach favors substrate nodes with lower failure probabilites. Meanwhile, for the case where $v_v$ is not the first node but also has no mapped neighboring VN nodes, the cost value for a candidate node $v_s \in \mathbb{L}$, $c_{v_s}$, is instead set to a weighted sum of the risk value, $r_{v_s}$, and total static hop count from $v_s$ to all other substrate nodes $v'_s$ assigned to already-mapped VN nodes in the request. The goal here is to map $v_v$ close to other mapped (non-neighboring) nodes, even if none of its VN links will be mapped in the current iteration. Carefully note that a weighting fraction, $\theta$, is also used to adjust the balance between risk and hop counts, see Steps 14-15 in Figure 4.3 (typically set to 1). Based upon the above, the substrate node, $v_s$, with the minimum $c_{v_s}$ value is chosen as the mapping for $v_v$, i.e., Step 24 in Figure 4.3. Finally, for the case where $v_v$ already has mapped neighbor VN node(s), the cost value for a node $v_s \in \mathbb{L}$, $c_{v_s}$, is assigned as a weighted sum of risk value, $r_{v_s}$, and total path cost from $v_s$ to all other nodes $v'_s$ corresponding to mapped neighboring VN nodes of $v_v$. Again, $\theta'$ is another fixed fraction used to adjust the balance between

1: Initial risk value for each substrate node and link, $r_{v_s} = \vec{\boldsymbol{p}}_{v_s} \cdot \vec{\boldsymbol{w}}$, $r_{e_s} = \vec{\boldsymbol{p}}_{e_s} \cdot \vec{\boldsymbol{w}}$
2: Sort VN nodes in descending order of node degree
3: **for** each $v_v \in V_v$
4: $v_s$=Call subroutine FSN_RMM($v_v$), if success reserve node resource;otherwise return FAIL
5: $r_{v_s} = 0$
6: **for** each adjacent VN link $(v_v, v'_v)$
7: **if** $v'_v$ is mapped
8: Compute minimum-cost path $\mathbb{P}_{(v_v,v'_v)}$ by using risk values as link weights, if success reserve bandwidth resource; otherwise return FAIL
9: $r_{v_s} = 0, \forall v_s \in \mathbb{P}_{(v_v,v'_v)}$, $r_{e_s} = 0, \forall e_s \in \mathbb{P}_{(v_v,v'_v)}$

Figure 4.2: Risk minimization mapping (RMM) algorithm

risk and path cost, see Steps 17-20 in Figure 4.3 (typically set to 1). Based upon the above, the substrate node $v_s$ with the minimum $c_{v_s}$ value is selected to map $v_v$, see Step 24 in Figure 4.3.

Overall, the complexity of the above scheme can be derived by analysing the individual components of the RMM algorithm in Figure 4.2. Namely, the sorting procedure (Step 2 Figure 4.2) has $O(|V_v||log|V_v|)$ complexity. Meanwhile, the complexity of iterating and mapping each VN node is determined by the FSM_RMM subroutine. Now the complexity of this particular subroutine is dominated by the candidate substrate node selection steps, i.e., Steps 10-20, Figure 4.3. Hence this yields a complexity of $O(|V_s||E_s|log|V_s|)$ for the FSM_RMM subroutine and a total algorithm complexity of $O(|E_v||V_s||E_s|log|V_s|)$ for the RMM scheme.

### 4.2.2 VN Embedding with Joint RM and TE

Overall, the RMM embedding scheme primarily tries to reduce the failure probability of a computed VN mapping. As such this approach ignores the revenue and cost

Input VN node $v_v$
**for** each $v_s \in V_s$
  **if** $v_s$ is not assigned
    $b_{max}$=maximum bandwidth among adjacent VN links of $v_v$
    $B_{max}$=maximum bandwidth among adjacent substrate links of $v_s$
    $b_{total}$=total bandwidth of adjacent VN links of $v_v$
    $B_{total}$=total bandwidth of adjacent substrate links of $v_s$
    **if** $R(v_s) \geq r(v_v)$ and $B_{max} \geq b_{max}$ and $B_{total} \geq b_{total}$
      Add $v_s$ into candidate substrate node list $\mathbb{L}$
**for** each $v_s \in \mathbb{L}$
  **if** $v_v$ is the first VN node
    $c_{v_s} = r_{v_s}$
  **else if** $v_v$ has no mapped neighbor VN nodes
    $hc$=total hop count from $v_s$ to $v'_s$ for all $v'_s$ are mapped
    $c_{v_s} = \theta * r_{v_s} + hc$
  **else**
    $mc = 0$
    **for** each mapped neighbor VN nodes $v'_v$
      Compute minimum-cost path $\mathbb{P}_{(v_v,v'_v)}$ using risk values as link weights, if success the path cost is added to $mc$; otherwise $mc = \infty$
    $c_{v_s} = \theta' * r_{v_s} + mc$
**if** $v_v$ is the first VN node
  Randomly select a $v_s \in \mathbb{L}$ with probability inversely proportional to $c_{v_s}$
**else**
  Select the $v_s \in \mathbb{L}$ with the minimum $c_{v_s}$ if $c_{v_s} \neq \infty$; otherwise return FAIL

Figure 4.3: Subroutine FSN_RMM algorithm

aspects for embedding design, and this can lead to increased resource inefficiency and lower revenues for cloud service providers. Hence in order to address this shortcoming, a novel scheme is proposed to take into account both risk minimization and TE resource efficiency concerns, termed as the *joint risk minimization and traffic engineering mapping* (JRTM) scheme.

The proposed JRTM scheme is detailed in Figure 4.4. First, a special data set $\mathbb{S}$ is used to record all of the "used" substrate nodes and links for an incoming request. Namely, $v_s \in \mathbb{S}$ if and only if $v_s$ is assigned to a VN node or it is an intermediate

node of a VN link. Similarly, $e_s \in \mathbb{S}$ if and only if $e_s$ is assigned to a VN link. Now akin to the RMM scheme, this algorithm also starts out by sorting each VN node in descending order of node degree. It then loops through all of the VN nodes and iteratively tries to map each to a substrate node, i.e., call to *FSN_JRTM* routine (Figure 4.5). Now pending a successful mapping here, the VN link connections are also routed (for adjacently-mapped VN nodes only). In particular, connection routes are computed by first resolving the $k$-shortest paths between the respective substrate nodes. A risk value, $r_{\mathbb{P}}$, is then computed for each of these $k$ paths, $\mathbb{P}$, as follows:

$$r_{\mathbb{P}} = \sum_{i=1}^{|U|} w_{u^i}(1 - \prod_{v_s \in \mathbb{P}, v_s \notin \mathbb{S}} (1 - p_{v_s}^{u^i}) \prod_{e_s \in \mathbb{P}, e_s \notin \mathbb{S}} (1 - p_{e_s}^{u^i})) \qquad \text{(Eq. 4-5)}$$

$$r_{v_s} = \begin{cases} \vec{\boldsymbol{p}}_{v_s} \cdot \vec{\boldsymbol{w}} & \text{if } v_s \notin \mathbb{S} \\ 0 & \text{if } v_s \in \mathbb{S} \end{cases} \qquad \text{(Eq. 4-6)}$$

Specifically, this risk value is the failure probability of path $\mathbb{P}$ under all potentail stressors. Note that, if a substrated node (link) is marked as "used" in $\mathbb{S}$, i.e., $v_s \in \mathbb{S}$ ($e_s \in \mathbb{S}$), then its failure probability is set to zero and it is precluded from the computation. Again, the reason here is that re-using already-assigned nodes (links) in the same VN will not introduce additional risk. Finally the path with the minimum risk value is selected for mapping the VN link. Now after a VN node has been mapped, the substrate node that it is mapped to is also marked as "used" in $\mathbb{S}$, see Step 5 in Figure 4.4. Similarly, after a VN link has been mapped, all the substrate nodes and links along its path route are also marked as "used" in $\mathbb{S}$, see Steps 12-13 in Figure 4.4.

Next, the joint risk and TE-based VN node mapping scheme is detailed in Figure 4.5, i.e., pseudocode for FSN_JRTM subroutine. Akin to the FSN_RMM algorithm in Figure 4.3, this algorithm also starts by building a candidate substrate node list, $\mathbb{L}$, for VN node $v_v$ based upon the same three criteria. Next, all candidate substrate

nodes in list $\mathbb{L}$ are evaluated to select one for mapping based upon three possible cases. In the first case, if $v_v$ is the first VN node in the VN request, its cost value $c^1_{v_s}$ is set to $\mathbb{C}(v_s)r(v_v)$ for each $v_s \in \mathbb{L}$. At the same time another cost value, $c^2_{v_s}$, is also defined as the risk value of $v_s$, i.e., $r_{v_s}$, as computed in Eq. 4-6. A further subset list $\mathbb{L}'$ is then built by randomly selecting $\lceil \frac{|\mathbb{L}|}{k'} \rceil$ substrate nodes from $\mathbb{L}$ with selection probabilities set as inversely-proportional to the node $c^1_{v_s}$ values. The final node mapping is then chosen from this reduced list by choosing a node $v_s$ from $\mathbb{L}'$ with the selection probabilities set as inversely-proportional to the $c^2_{v_s}$ values, see Steps 26-27 in Figure 4.5. Overall, this "two-stage" selection process first tries to evenly-distribute the (first) VN node over the substrate network and then incorporates failure risks to minimize its placement risk, i.e., jointly lowering mapping costs and reducing risk at the same time.

Next consider the case where VN node $v_v$ is not the first node, but also has no mapped neighboring VN nodes. Here the cost for a substrate node $v_s$, $c^1_{v_s}$, is set to a weighted sum of $\mathbb{C}(v_s)r(v_v)$ and total static hop counts from $v_s$ to all other substrate nodes $v'_s$ assigned to already-mapped VN nodes in the VN request. The reason here is as same as in FSN_RMM subroutine (as discussed in Section 4.2.1). In addition, $c^2_{v_s}$ is also set to $r_{v_s}$ as computed using Eq. 4-6. Based upon the above, the algorithm sorts the substrate nodes in $\mathbb{L}$ by ascending $c^1_{v_s}$ values and also generates a further subset list $\mathbb{L}'$ by selecting the first $\lceil \frac{|\mathbb{L}|}{k'} \rceil$ substrate nodes from $\mathbb{L}$. From this reduced list, the substrate node $v_s$ with the minimum $c^2_{v_s}$ value is then selected as the mapping for $v_v$, see Steps 29-31 in Figure 4.5. Again, this two-stage selection incorporates both risk minimization and TE efficiency concerns. Finally, for the case where $v_v$ already has some mapped neighbor VN node(s), the cost values $c^1_{v_s}$ are computed as a weighted sum of $\mathbb{C}(v_s)r(v_v)$ and total path cost from $v_s$ to all other nodes $v'_s$ corresponding to mapped neighboring VN nodes of $v_v$. Meanwhile, the $c^2_{v_s}$ values are set to the weighted sum of $r_{v_s}$ and the total risk value of the paths computed by Eq. 4-5. Finally, the subsequent process for selecting the final candidate substrate node

```
1: Initial track "used" substrate node and link set 𝕊 = ∅
2: Sort VN nodes in descending order of node degree
3: for each v_v ∈ V_v
4:    v_s=Call subroutine FSN_JRTM(v_v), if success reserve node re-
      source;otherwise return FAIL
5:    Add v_s to 𝕊
6:    for each adjacent VN link (v_v, v'_v)
7:       if v'_v is mapped
8:          Compute k-shortest path for (v_v, v'_v) by using link cost values
            as link weights, if no path computed successfully return FAIL
9:          for each path ℙ of k paths
10:            Compute risk value r_ℙ for ℙ by Eq. 4-5
11:         Select ℙ with minimum r_ℙ, reserve bandwidth resource
12:         Add v_s, ∀v_s ∈ ℙ to 𝕊
13:         Add e_s, ∀e_s ∈ ℙ to 𝕊
```

Figure 4.4: Joint risk minimization and TE mapping (JRTM) algorithm

is the same as that for the second case, i.e., node $v_v$ without any mapped neighbor.

The overall complexity of the JRTM scheme can also be analysed in a similar manner to that for the RMM scheme in Section 4.2.1. Namely, the complexity of FSN_JRTM subroutine is bounded by $O(|V_s||E_s|log|V_s|)$. Hence the total complexity of JRTM scheme is $O(|E_v|(|V_s|+k)|E_s|log|V_s|)$.

Input VN node $v_v$
**for** each $v_s \in V_s$
  **if** $v_s$ is not assigned
    $b_{max}$=maximum bandwidth among adjacent VN links of $v_v$
    $B_{max}$=maximum bandwidth among adjacent substrate links of $v_s$
    $b_{total}$=total bandwidth of adjacent VN links of $v_v$
    $B_{total}$=total bandwidth of adjacent substrate links of $v_s$
    **if** $R(v_s) \geq r(v_v)$ and $B_{max} \geq b_{max}$ and $B_{total} \geq b_{total}$
      Add $v_s$ into candidate substrate node list $\mathbb{L}$
**for** each $v_s \in \mathbb{L}$
  **if** $v_v$ is the first VN node
    $c^1_{v_s} = \mathbb{C}(v_s)r(v_v)$
    $c^2_{v_s} = r_{v_s}$, computed by Eq. 4-6
  **else if** $v_v$ has no mapped neighbor VN nodes
    $hc$=total hop count from $v_s$ to $v'_s$ for all $v'_s$ are mapped
    $c^1_{v_s} = \theta * \mathbb{C}(v_s)r(v_v) + hc$
    $c^2_{v_s} = r_{v_s}$, computed by Eq. 4-6
  **else**
    $mc = 0$, $rc = 0$
    **for** each mapped neighbor VN nodes $v'_v$
      Compute minimum-cost path $\mathbb{P}$ for $(v_v, v'_v)$ by using link cost as link weights, if success the path cost is added to $mc$; otherwise $mc = \infty$
      Compute risk value $r_{\mathbb{P}}$ for $\mathbb{P}$ by Eq. 4-5 and added it to $rc$
    $c^1_{v_s} = \theta' * \mathbb{C}(v_s)r(v_v) + mc$
    $c^2_{v_s} = \theta' * r_{v_s} + rc$, $r_{v_s}$ is computed by Eq. 4-6
**if** $v_v$ is the first VN node
  Build $\mathbb{L}'$ as randomly select $\lceil \frac{|\mathbb{L}|}{k'} \rceil$ substrate nodes from $\mathbb{L}$ with probability inversely proportional to $c^1_{v_s}$
  Randomly select $v_s \in \mathbb{L}'$ with probability inversely proportional to $c^2_{v_s}$
**else**
  If $c^1_{v_s} = \infty, \forall v_s \in \mathbb{L}$, return FAIL; otherwise, sort $\mathbb{L}$ in ascending order according to $c^1_{v_s}$
  Select the first $\lceil \frac{|\mathbb{L}|}{k'} \rceil$ substrate nodes in $\mathbb{L}$ to build $\mathbb{L}'$
  Select the $v_s \in \mathbb{L}'$ with the minimum $c^2_{v_s}$

Figure 4.5: Subroutine FSN_JRTM algorithm

## 4.3 Performance Evaluation

The "risk-aware" VN mapping schemes are now tested using $OPNETModeler^{TM}$ for the same 24-node network with 5 pre-defined failure regions. This network is re-drawn in Figure 4.6 to indicate the failure probability for each stressor, i.e., $\vec{\boldsymbol{w}}$. In addition, some sample risk vectors are also illustrated in this diagram. Again, all substrate nodes have 100 units of capacity and all substrate links have 10,000 units of bandwidth. Client user VN requests are also varied between 4-7 nodes, and average VN node degrees are set to 2.6, i.e., computed as the ratio of VN links to VN nodes. Also, the average requested VN node capacity is uniformly distributed between 1-10 units, and the average requested VN link capacity is uniformly distributed between 50-1,000 units. As per Chapter 3, all requests have exponential holding and inter-arrival times, with means $\mu$ and $\lambda$, respectively. Namely, a value of $\mu = 600$ time units is used and $\lambda$ is varied per load.

Furthermore, all tests are done with 100,000 randomly-generated VN requests, and multi-failure stressor events are randomly triggered after about 1,000 requests. Specifically, when a multi-failure event occurs, a random stressor is chosen from the set $U$ according to the pre-defined occurrence probabilities, $\vec{\boldsymbol{w}} = [0.2, 0.1, 0.1, 0.2, 0.4]$. Next, each substrate node and link (in the failure region of this stressor) is failed independently as per its pre-defined probability values. For comparison purposes, the regular "non-risk" NSVIM mapping scheme is also tested in the simulations (see Appendix A.1). Detailed results and findings are now presented.

### 4.3.1 Blocking Rates

Initial tests are done to measure VN request blocking rates for the various schemes as shown in Figure 4.7. Overall, these findings show that the base NSVIM scheme yields the lowest blocking, whereas the RMM scheme gives the highest blocking.

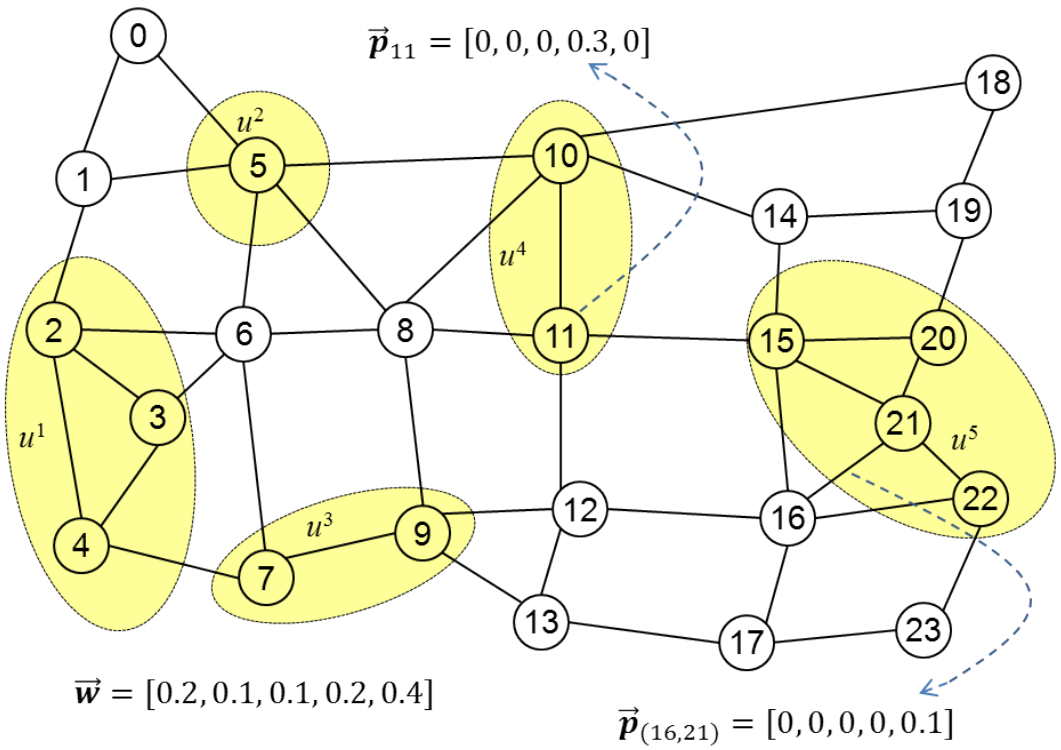



Figure 4.6: 24-node substrate network with probabilistic stressors

This is expected since the NSVIM scheme is a pure "TE-only" solution which tries to minimize VN resource usage. By constrast, the RMM scheme does not incorporate TE concerns and chooses longer and more circuitous connection routes for the VN links. Furthermore, the joint JRTM heuristic also achieves a very good tradeoff between the above two alternatives. For example this scheme closely tracks the NSVIM scheme at medium-high loads (with 1.2 times higher blocking) and notably outperforms the RMM scheme at all loads (by almost an order of magnitude lower blocking).

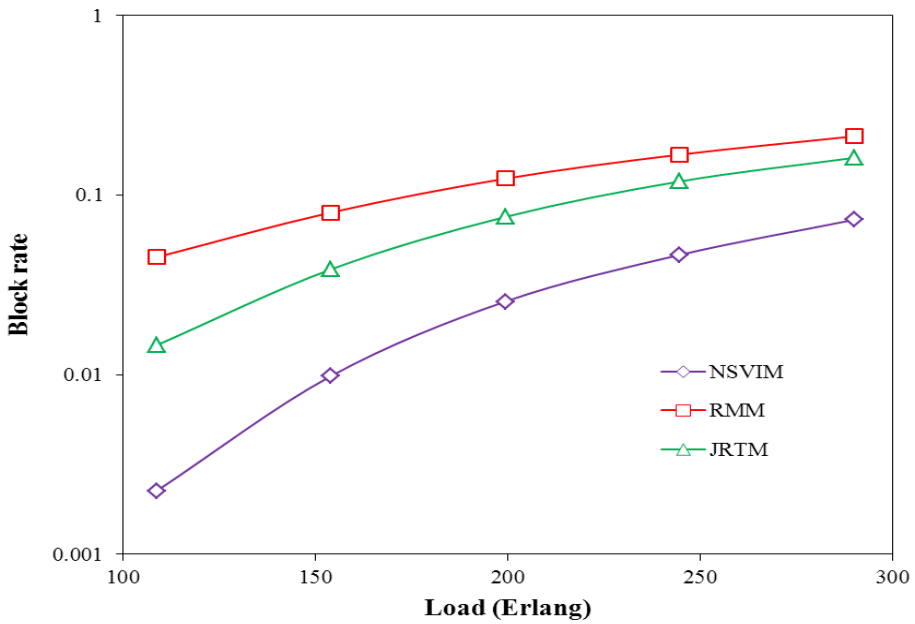



Figure 4.7: Blocking rate

### 4.3.2 Net Revenue

The net revenues for the various heuristics are also plotted in Figure 4.8. These results indicate that the NSVIM scheme yields the highest net revenue, whereas, again the RMM scheme gives the lowest results. As expected, the JRTM solution comes in between these two, achieving about 24% lower net revenue than the NSVIM scheme at high loads, but 22% higher revenue than the RMM scheme. Overall, this improved performance with NSVIM and JRTM schemes is due to the fact they are more efficient in terms of resource utilization.

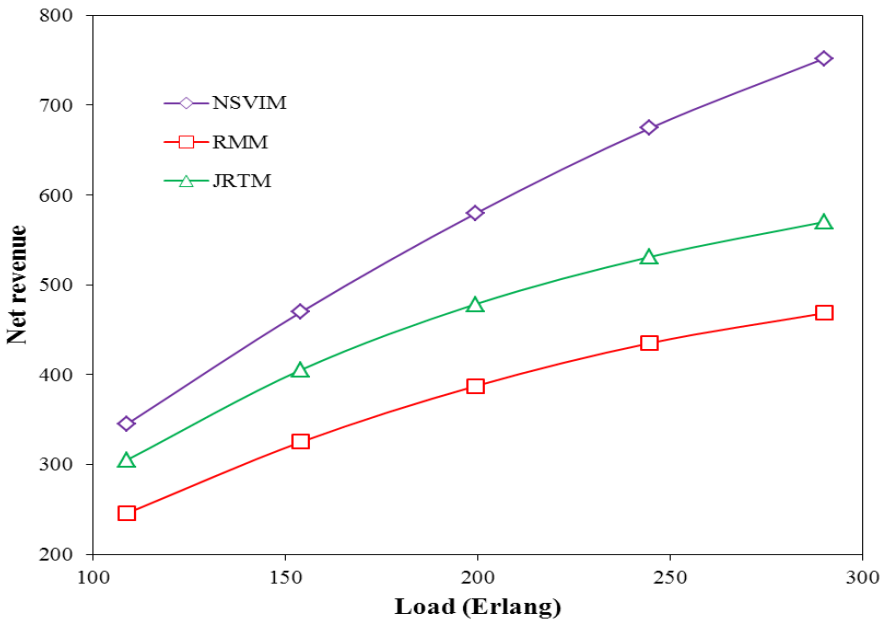



Figure 4.8: Net revenue

### 4.3.3 Number of Failed VN

Next, the number of failed VN requests is plotted in Figure 4.9, i.e., VN requests experiencing at least one or more VN node and/or link failure. These findings show that NSVIM scheme gives the worst reliability of all, i.e., averaging 53% (46%) more failures than the RMM (JRTM) scheme at high loads. More importantly, the JRTM scheme is extremely effective and closely tracks the performance of the pure risk-based RMM solution to within 8% of failed VN requests.

### 4.3.4 Failure Rate

Finally, the failure rates of the various schemes are also analyzed here. In particular, the failure rate is defined as the ratio of total affected VN requests to all accepted

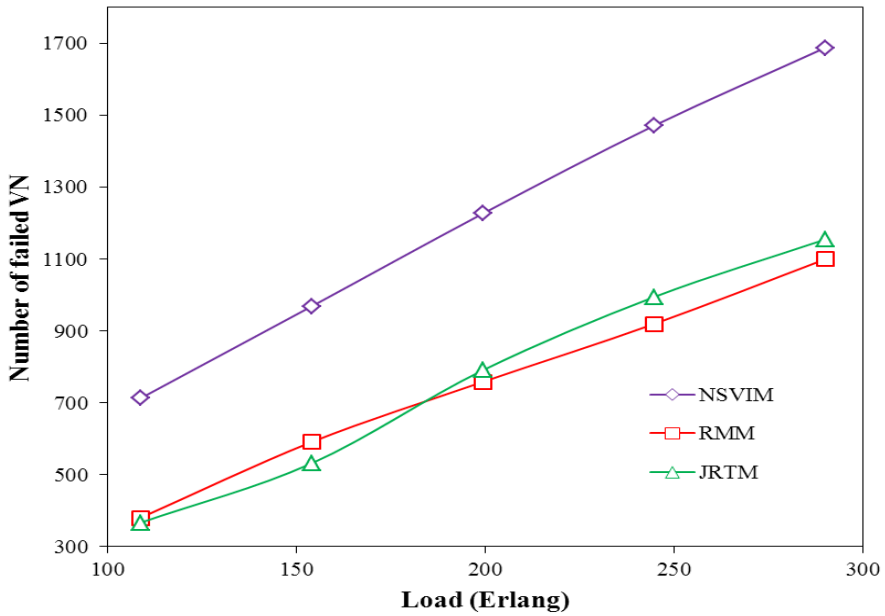



Figure 4.9: Number of failed VN

VN requests. The corresponding results are plotted in Figure 4.10 and show that the NSVIM scheme yields the highest failure rate, i.e., as it ignores risk minimization concerns. By contrast, both the RMM and JRTM schemes achieve nearly the same level of performance, i.e., 48% lower than NSVIM at low loads. This clearly shows the benefit of incorporating risk minimization information into the VN mapping process.

Also note that the RMM scheme gives slightly higher failure rates than the JRTM scheme at low loads, i.e., by about 16%. This increase is due to the fact that the latter scheme gives lower blocking rates than the RMM solution. Hence when the total number of VN failures is similar, the larger numbers of accepted VN requests imply lower *relative* failure rates for the JRTM scheme. This difference is also shown more clearly in Figure 4.11, which plots the ratio of total affected VN requests to the total number of active VN requests during a failure event. Here the RMM scheme gives up to 18% higher ratios than the JRTM scheme. These results show

that jointly considering risk minimization and TE concerns achieves better failure avoidance than only focusing on risk, i.e., as the former approach gives much lower blocking and similar numbers of VN failures.

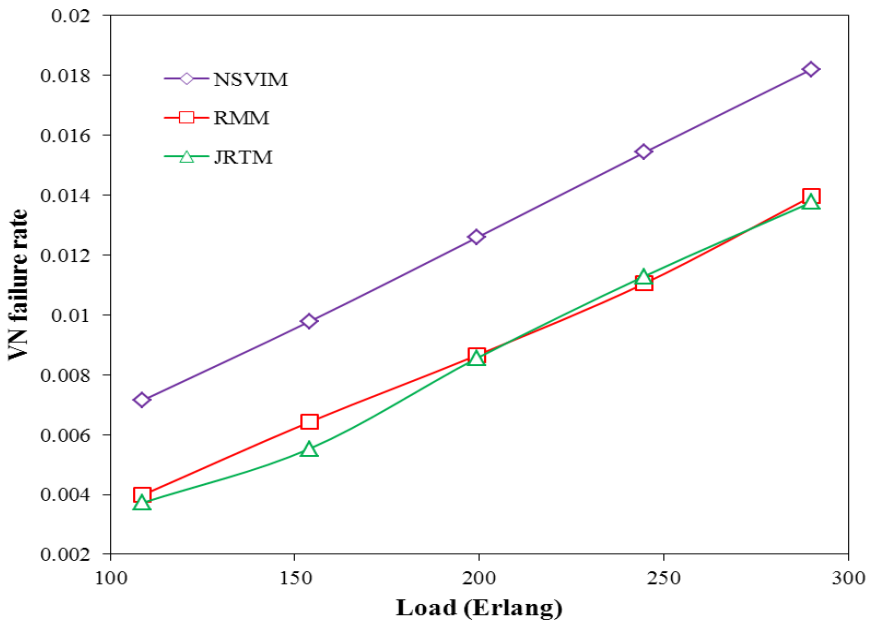



Figure 4.10: VN failure rate

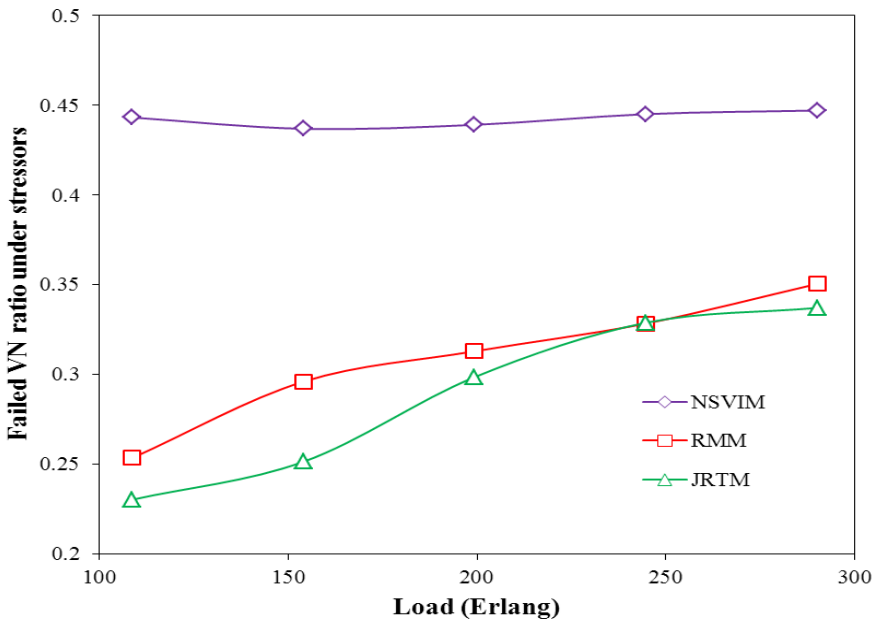



Figure 4.11: Failed VN ratio under stressors

# Chapter 5

# Post-Fault VN Restoration

In general, pre-provisioned protection schemes entail higher resource consumption since backup resources must be reserved in advance. By contrast, restoration-based strategies only provision working resources and use *post-fault* recovery strategies to restore entities after failures. Although these schemes yield longer delays, they can provide much better resource efficiency. Furthermore, restoration-based strategies are also more dynamic and therefore very attractive in multi-failure settings, i.e., as it is very difficult to pre-provision protection resources for all node and link failure combinations.

Now the only known work on post-fault VN restoration is presented in [MR01]. Namely, here the authors pre-partition link bandwidth into two groups, i.e., one for working VN mappings and the other for backup post-failure VN mappings. However, since resources are reserved in advance, this scheme still yields relatively high resource consumption, akin to pre-computed protection. Moreover, it only considers single link failure events and is unable to handle wider regional stressors. In general, these larger-scale failure events will impact many more active VN requests, as multiple substrate nodes/links may go down. Hence in order to restore these affected

requests, two approaches can be considered. Namely, either the whole affected VN can be migrated to working nodes/links or only its failed portion can be re-mapped. However, due to limited substrate network resources after a failure, it may not be possible to fully recover all failed requests. Regardless, it may still be beneficial to restore part of an affected VN in order to provide a degraded/partial level of service. This approach may also alleviate carrier concerns about service interruptions during post-fault VN migration.

Along these lines, this chapter motivates more efficient post-fault restoration strategies for survivable VN mapping under regional multi-failure events. In particular, three solutions are presented here, with the first remapping each affected VN to a new location and the latter two only restoring the failed VN nodes/links. Consider the details.

## 5.1 Network Model and Description

The overall notation and network model for VN restoration is similar to that introduced in Section 3.1. Furthermore, many of the performance evaluation metrics defined for VN protection are also applicable here. However, some further definitions are also needed to address more specialized post-fault VN restoration needs. For example, the penalty associated with service disruption is defined as:

$$PE(G_v) = \sum_{e_f \in E_f} b(e_f) * \mathbb{P}(e_f) + \psi \sum_{v_f \in V_f} r(v_f) * \mathbb{P}(v_f) \qquad \text{(Eq. 5-1)}$$

where $V_f \subseteq V_v$ is the set of failed VN nodes, $E_f \subseteq E_v$ is the set of failed VN links, $\mathbb{P}(e_f)$ is the unit link penalty, $\mathbb{P}(v_f)$ is the unit node penalty, and $\psi$ is the fraction of node penalty. Using the above, the net revenue is also re-defined as:

$$\frac{\sum_i (REV(G_v^i) - COST(G_v^i)) - \sum_j PE(G_v^j)}{T}, \forall G_v^i \in \mathbb{A}, \forall G_v^j \in \mathbb{F} \qquad \text{(Eq. 5-2)}$$

where $REV(G_v)$ and $COST(G_v)$ are defined in Eq. 3-1 and Eq. 3-2, respectively, $G_v^i$ is the *i-th* VN request, $\mathbb{A}$ is the set of all accepted VN requests, $G_v^j$ is the *j-th* VN request, $\mathbb{F}$ is the set of all failed VN requests, and $T$ is the total run time. Furthermore, a VN "completeness ratio" is also defined as:

$$\omega * (1 - \frac{|V_f|}{|V_v|}) + (1 - \omega) * (1 - \frac{|E_f|}{|E_v|}) \quad \text{(Eq. 5-3)}$$

where $\omega$ $(1 - \omega)$ is the fraction of node (link) completeness. Specifically, this ratio measures the proportion of a VN mapping that cannot be recovered. Finally, the overhead cost for restoring a failed VN request is also defined as:

$$\chi * |V_r| + |E_r| \quad \text{(Eq. 5-4)}$$

where $V_r$ is the set of VN nodes that change their locations (mappings) in the substrate network after restoration, and $E_r$ is the set of VN links that alter their substrate connection paths after restoration. Similarly, $\chi$ is the fraction of node restoration overhead.

## 5.2 VN Post-fault Restoration

Some new post-fault VN restoration schemes are now proposed. Overall, these algorithms use a greedy approach and either migrate the entire (affected) VN to a new location or only restore its failed VN nodes and links. Consider the details.

### 5.2.1 Full Migration Restoration (FMR)

The *full migration restoration* (FMR) scheme re-computes a new VN mapping for each affected VN request and is detailed in Figure 5.1. If this computation is successful, the entire VN request is migrated to the new node/link locations. Otherwise the

1: Build VN set $F$ by grouping all affected VN requests
2: Sort VN requests in $F$ in ascending order of the number of affected VN nodes/links
3: **for** each $G_v \in F$
4: $M_{prev}$=record current node/link mapping of $G_v$
5: Release resources for $G_v$
6: $M_{new}$=compute VN mapping for $G_v$
7: **if** $SUCCESS$
8: Reserve resource according to $M_{new}$
9: **else**
10: Reserve resource according to $M_{prev}$

Figure 5.1: Full migration restoration (FMR) algorithm

VN request retains its original mapping, with the intact non-failed portions providing a reduced level of user service. Now the overall algorithm starts by first identifying and grouping all failed VN requests into a set $\boldsymbol{F}$. These requests are then sorted according to their number of affected VN nodes/links in ascending order. The goal here is to restore VN requests with fewer numbers of affected VN nodes/links first. Next, a new mapping is computed for each request in $\boldsymbol{F}$ using any regular VN mapping algorithm. Based upon the result of this re-computation, the VN request is either migrated to its new mapping or left in place. Note that if the VN request is migrated then even working (non-failed) nodes/links must be migrated, and this may lead to increased overheads, costs, and service interruptions.

### 5.2.2 Partial Migration Restoration (PMR)

In general the FMR algorithm will yield higher overheads for operators. As a result, a more efficient *partial migration restoration* (PMR) scheme is also proposed, see Figure 5.2. Namely, this algorithm only restores affected VN nodes/links and does not re-map the intact parts of an affected VN request. Again, this scheme starts by grouping the affected VNs into a set $\boldsymbol{F}$. Next for each affected VN request, the

scheme identifies the subset of affected VN nodes, $V_f$, and VN links, $E_f$. A candidate substrate node list, $\boldsymbol{L}$, is then built for each VN node, $v_v \in V_f$, (see Step 5, Figure 5.2). Now several criteria are used for selecting these nodes. Foremost, a candidate substrate node cannot be allocated to any other VN node in the same VN request. Also, the available node resources, $R(v_s)$, at a candidate node $v_s$ must be greater than or equal to the requested amount, $r(v_v)$. Finally, the maximum available bandwidth on all adjacent substrate links of the candidate node, $v_s$, must not be less than $b_{max}(v_v)$, i.e., where $b_{max}(v_v)$ is the maximum bandwidth requirement between $v_v$ and the set of all VN nodes that are adjacent to $v_v$ in $V_f$.

Using the above, each VN node in $V_f$ is sorted according to the number of candidate substrate nodes in ascending order. The goal here is to first restore VN nodes with fewer candidate nodes, i.e., less mapping choices. Next, each VN node is mapped to one substrate node in its candidate node list, $\boldsymbol{L}$, based upon two criteria (i.e., Steps 7-10, Figure 5.2). First, the substrate node cannot be allocated to another VN node in the same VN request. Second, the chosen VN node must be the one yielding the maximum product value of the sum of available bandwidth on all adjacent substrate links multiplied by the value of available residual node resource, i.e., Step 9, Figure 5.2. If no such substrate node can be found for a VN node, then this VN node cannot be restored and the algorithm simply jumps to the next affected VN node in the request, i.e., *partial* restoration support. Finally, after all VN nodes mapping have been attempted, the affected VN links are recomputed if both of their end-point VN nodes are intact and/or successfully restored (i.e., Steps 11-12, Figure 5.2). Similarly, if VN link computation fails, then the VN link is deemed as failed and the algorithm moves to restore the next affected VN link.

1: Build VN set $F$ by grouping all affected VN requests
2: Sort VN requests in $F$ in ascending order of the number of affected VN nodes/links
3: **for** each $G_v \in F$
4: Find affected VN nodes set $V_f \subseteq V_v$ and VN links set $E_f \subseteq E_v$
5: **for** each $v_v \in V_f$
6: Build candidate substrate node list $\boldsymbol{L}$ for $v_v$
7: Sort $V_f$ according $|\boldsymbol{L}|$ of each $v_v \in V_f$ in ascending order
8: **for** each $v_v \in V_f$
9: **for** each $v_s \in \boldsymbol{L}$
10: Remove it from $\boldsymbol{L}$ if it is already allocated to other VN nodes
11: **for** each $v_s \in \boldsymbol{L}$
12: $\sigma_{v_s} = R(v_s) * \sum_{v'_s} B(v_s, v'_s)$, where $v'_s \in adj(v_s)$
13: Map $v_v$ to the $v_s$ that has maximum $\sigma_{v_s}$
14: **for** each $e_v \equiv (v_v, v'_v) \in E_f$, i.e., $v_v$ and $v'_v$ are unaffected/restored
15: Compute path for $e_v$ and reserve bandwidth if path computation successes

Figure 5.2: Partial migration restoration (PMR) algorithm

### 5.2.3 Joint-Partial Migration Restoration (J-PMR)

Since the PMR scheme separately restores VN nodes and VN links, it may yield lower resource efficiency. Hence, a *joint-partial migration restoration* (J-PMR) scheme is also proposed. Namely, here the adjacent links for a restored VN node are immediately re-mapped before treating the next VN node akin to similar strategies for regular (non-survivable) VN mapping, see Section 2.1. Now the detailed J-PMR algorithm is shown in Figure 5.3 and starts by grouping and sorting all affected VN requests, i.e., akin to the FMR scheme. Next for each affected VN request, the algorithm identifies its failed VN nodes/links and first tries to restore only those affected VN links whose two end-point VN nodes are not affected, i.e., only intermediate hops are failed (see Steps 5-6, Figure 5.3). Subsequently, all of the affected VN nodes are

sorted according to their node degree in descending order, and a candidate substrate node list, $\boldsymbol{L}$, is generated for each, i.e., similar to the PMR scheme. Furthermore, for each substrate node $v_s \in \boldsymbol{L}$, a path cost $P_c$ is also computed for the best route between each pair of nodes ($v_s$, $v'_s$), i.e., where $v'_s$ is allocated to a VN node that is adjacent to $v_v$. Now if any path route computation fails, then the associated path cost $P_c$ is set to infinity. From this computation, the candidate node $v_s$ with the minimum total path cost, $\tau_{v_s}$, is selected for mapping the affected node, $v_v$. However, if all the $\tau_{v_s}$ values are equal to infinity for all candidate nodes, then the associated VN node $v_v$ is considered as unrestorable, and the algorithm simply moves to the next affected VN node, i.e., *partial* restoration support as well. Finally, if $v_v$ is re-mapped, then each VN link adjacent to $v_v$ is also rerouted (see Steps 17-18, Figure 5.3).

## 5.3 Performance Evaluation

The post-fault VN restoration schemes are also tested in $OPNETModeler^{TM}$ using the earlier-defined 24-node network with 5 failure regions, shown in Figure 3.2 (Section 3.1). Again, all substrate nodes have 100 units of capacity and all substrate links have 10,000 units of bandwidth. VN requests are also varied between 4-7 nodes and average VN node degrees are set to 2.6, i.e., computed as the ratio of VN links to VN nodes. Also, the average requested VN node capacity is uniformly distributed between 1-10 units, and the average requested VN link capacity is uniformly distributed between 50-1,000 units. All requests have exponential holding and inter-arrival times, with means $\mu$ and $\lambda$, respectively. Namely, a value of $\mu = 600$ time units is used and $\lambda$ is varied per load. Furthermore, all tests are done with 100,000 randomly-generated VN requests, and multi-failure stressor events are randomly triggered after about 1,000 requests.

Two base VN mapping schemes are also selected here for regular VN mapping

1: Build VN set $F$ by grouping all affected VN requests
2: Sort VN requests in F in ascending order of the number of affected VN nodes/links
3: **for** each $G_v \in F$
4: Find affected VN nodes set $V_f \subseteq V_v$ and VN links set $E_f \subseteq E_v$
5: Find subset $E_p \subseteq E_f$ such that $\forall(v_v, v'_v) \in E_p$: $v_v$, $v'_v$ are not affected
6: Compute path for $e_v \in E_p$, and reserve bandwidth if path computation is successful
7: Sort $V_f$ according to VN node degree in descending order
8: **for** each $v_v \in V_f$
9: Build candidate substrate node list $\boldsymbol{L}$ for $v_v$
10: **for** each $v_s \in \boldsymbol{L}$
11: $\tau_{v_s} = 0$
12: **for** each $v'_v \in adj(v_v)$
13: **if** $v'_v$ is unaffected or restored
14: $P_c$=path cost from $v_s$ to $v'_s$, and $v'_s$ is allocated to $v'_v$
15: $\tau_{v_s} = \tau_{v_s} + P_c$
16: Map $v_v$ to the $v_s$ that has minimum $\tau_{v_s}$ and $\tau_{v_s} \neq +\infty$
17: **for** each $v'_v \in adj(v_v)$ where $v'_v$ is unaffected or restored
18: Compute path for $(v_v, v'_v)$ and reserve bandwidth if path computation successes

Figure 5.3: Joint-partial migration restoration (J-PMR) algorithm

purposes, i.e., NSVIM in [HY04] and the *baseline VN embedding* (BVNE) scheme in [MY01]. Now since these algorithms will likely yield varying levels of performance, the restoration strategies are appropriately grouped according to their base VN mapping algorithm, i.e., NSVIM or BVNE. Finally, to gauge post-fault restoration against pre-computed protection, the FRGBM scheme from Section 3.3 is also tested here. This solution has been found to be more efficient than other existing VN protection schemes and also uses the NSVIM solution as its base VN mapping (although it can also be coupled with other mapping schemes such as BVNE).

### 5.3.1 Blocking Rates

Initial tests are done to measure VN request blocking rates, see Figure 5.4. Overall, the findings show that all restoration schemes yield nearly identical behaviors for the same baseline VN mapping strategy. Restoration also outperforms protection (FRGBM scheme) by a very wide margin, e.g., 57% (48%) lower blocking at low-medium loads if NSVIM (BVNE) is used as regular VN mapping. Clearly, this is due to the fact that the pre-provisioned FRGBM scheme consumes higher levels of resources.

### 5.3.2 Net Revenue

The net revenues are also plotted in Figure 5.5 and again show superior performance with the post-fault restoration schemes. For example, these strategies give almost 2.2 (4.3) times more revenue than FRGBM at higher loads if NSVIM (BVNE) is used for the regular VN mapping. Overall, these findings confirm that post-fault restoration can yield substantially higher efficiency than pre-provisioned protection. Furthermore, all three post-fault restoration schemes give nearly the same revenue levels for the same baseline VN mapping strategy. This behavior is due to the fact that the frequency (and duration) of failure events is relatively small as compared to the overall duration of VN requests. Hence, several additional metrics are also evaluated to provide a more detailed look at post-fault recovery performance.

### 5.3.3 VN Completeness Ratio

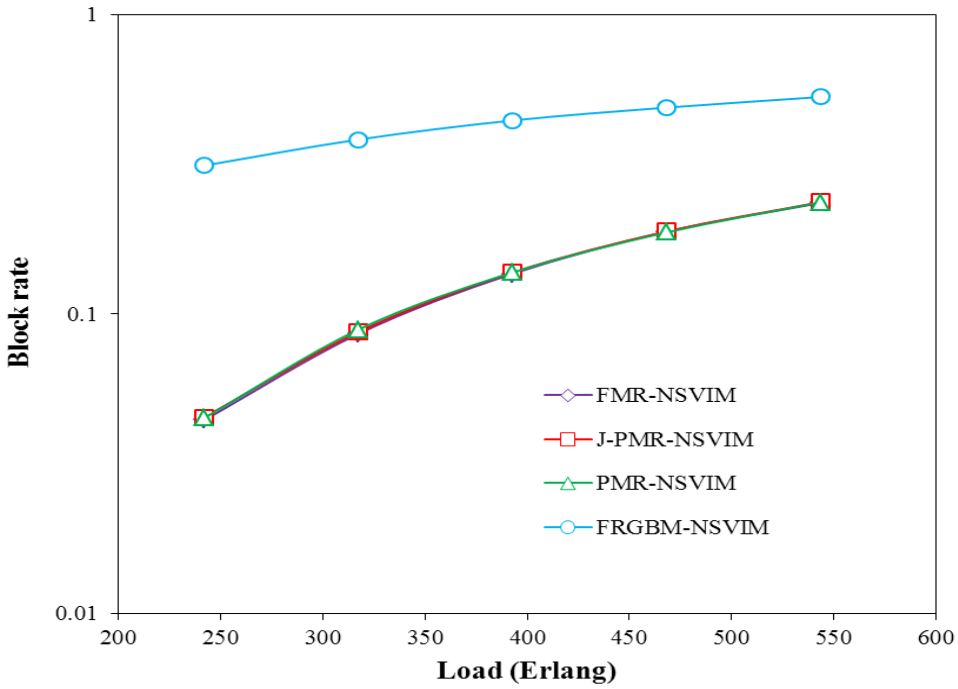
Average VN completeness ratios are plotted in Figure 5.6. These results show that the FMR scheme gives slightly higher recovery than J-PMR, i.e., about 4% more when using the NSVIM base VN mapping scheme. In general, this improvement is

due to the fact that the FMR scheme migrates the entire VN and hence has more flexibility in finding new working substrate locations. Additionally, the results also indicate that the J-PMR scheme gives higher completeness ratios than the PMR scheme, i.e., up to 13% (15%) more than PMR when using the NSVIM (BVNE) base VN mapping scheme. This is expected since joint VN node/link mapping is generally more resource efficient, leading to more successful recovery attempts.

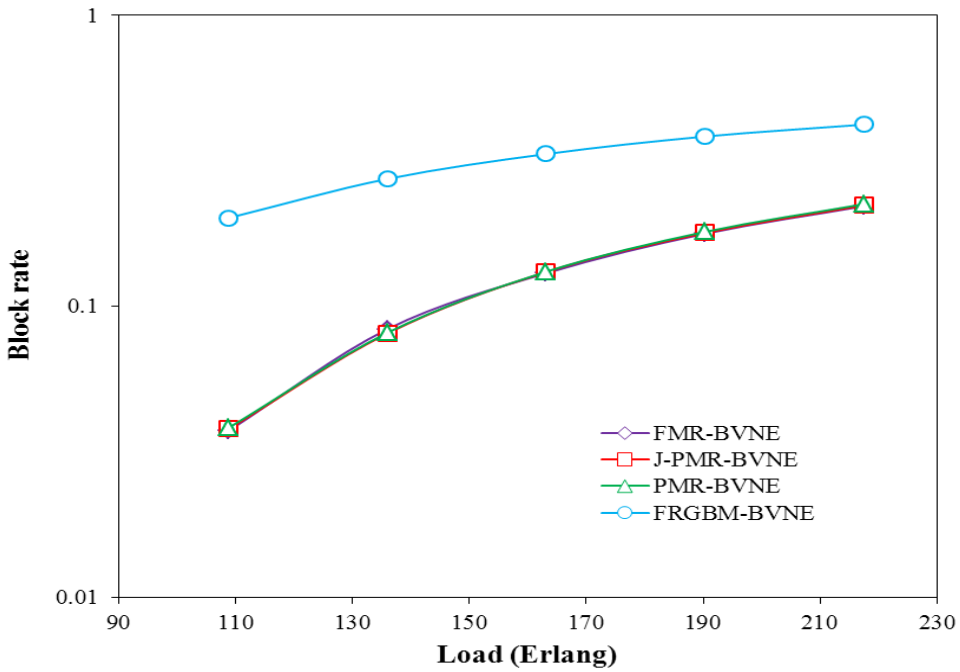
To further study restoration performance, the ratio of fully-restored VNs is also plotted in Figure 5.7. Here fully-restored implies that all affected VN nodes and links are recovered. Overall, these result show that the FMR scheme gives the highest restoration ratios, i.e., ranging from 0.65 to 0.8, due to full re-mapping. Meanwhile the J-PMR scheme also achieves much higher restoration ratios than the PMR scheme, i.e., up to 2.2 (1.0) times more than PMR at low loads when using the NSVIM (BVNE) base VN mapping scheme. Clearly, the J-PMR algorithm also benefits from joint node/link re-mapping procedures.

### 5.3.4 Restoration Overhead

Finally, average VN restoration overheads are plotted in Figure 5.8. Here the partial re-mapping PMR and J-PMR schemes yield much lower overheads as compared to the full-remapping FMR solution, i.e., up to 79% (89%) lower overheads than FMR for J-PMR (PMR) when using the NSVIM base VN mapping scheme. This is expected since both the J-PMR and PMR schemes only implement partial recovery of affected VN nodes and links.

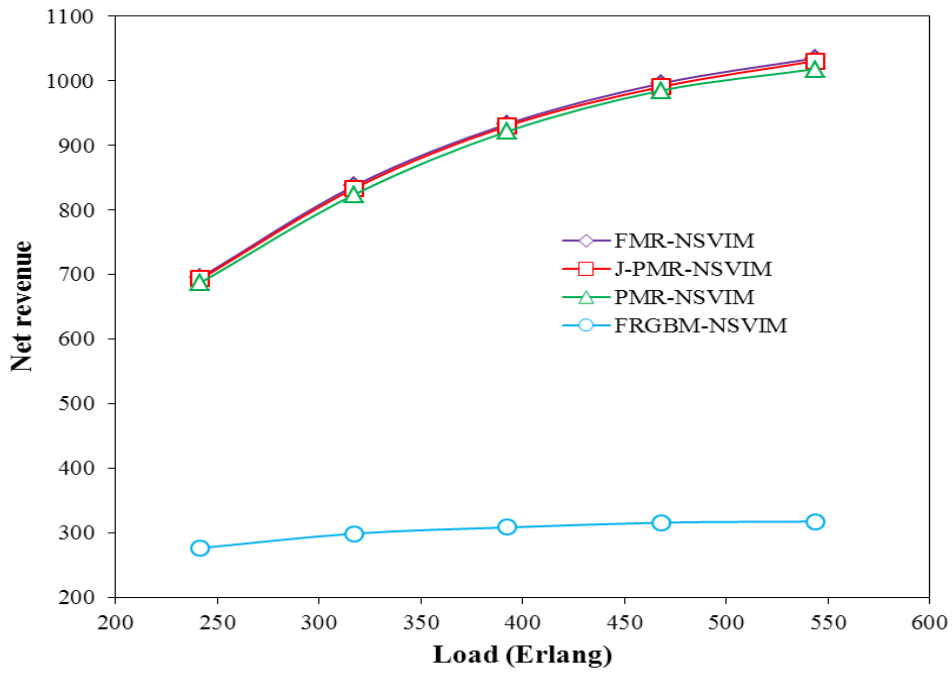



(a)

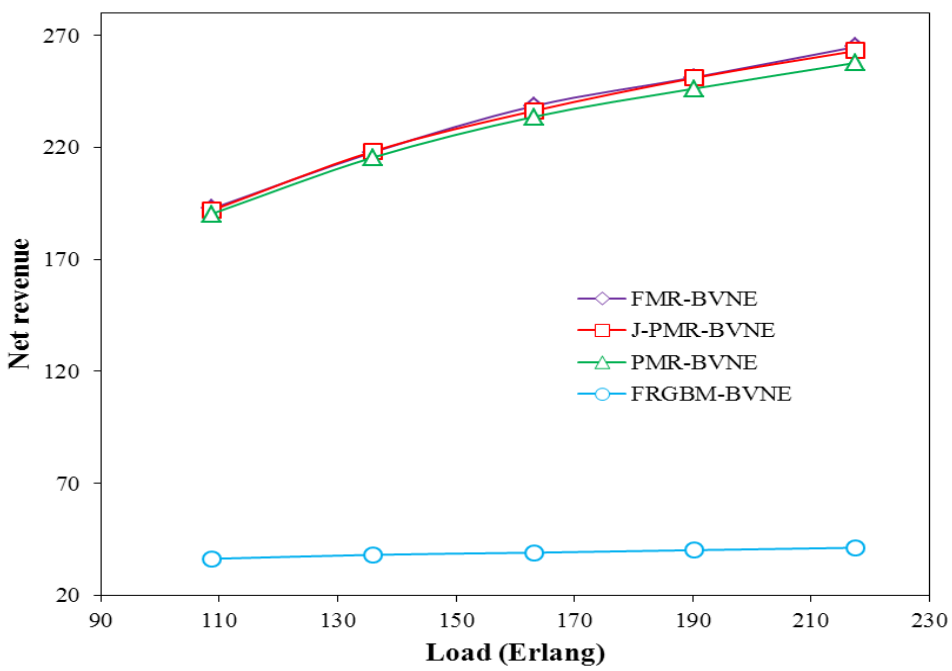



(b)

Figure 5.4: Blocking rate: a) NSVIM b) BVNE

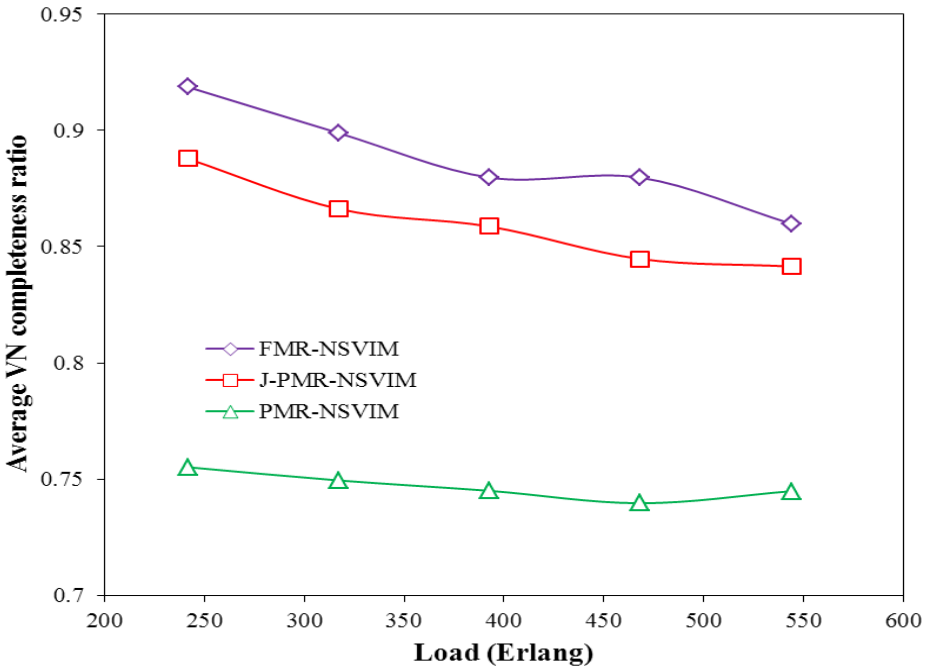



(a)

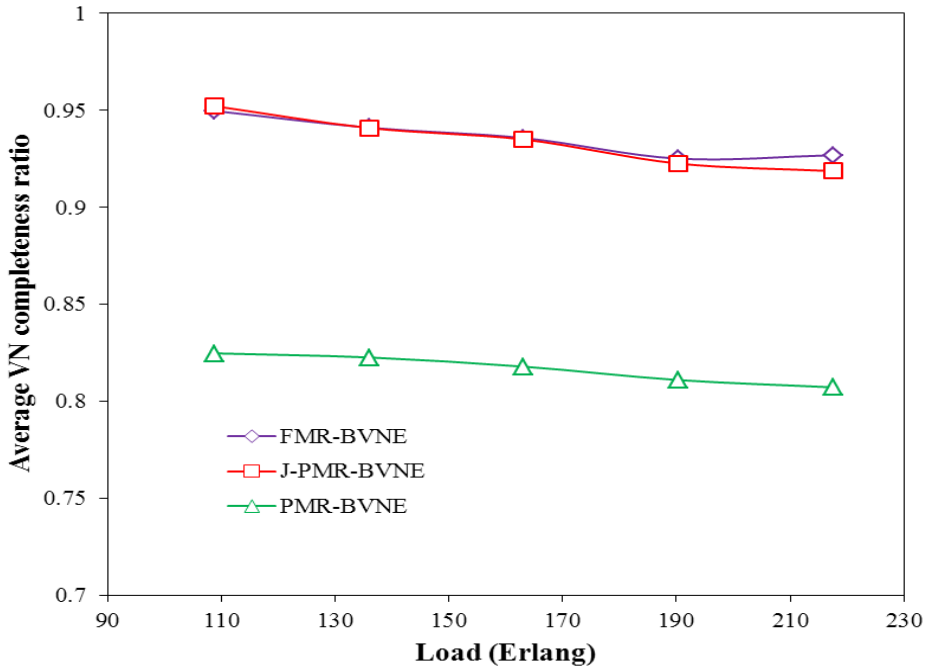



(b)

Figure 5.5: Net revenue: a) NSVIM b) BVNE

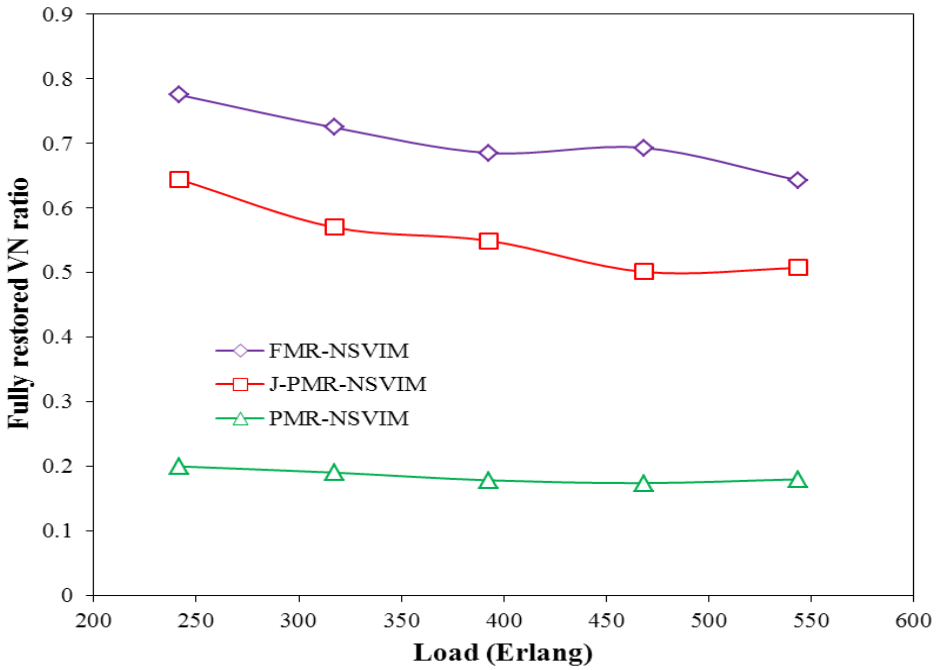



(a)

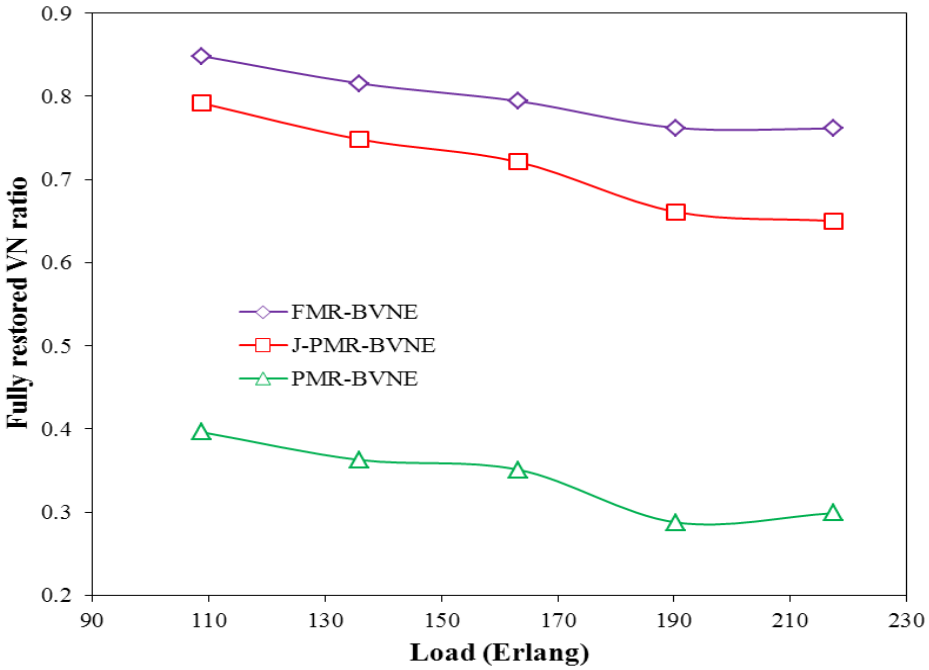



(b)

Figure 5.6: VN completeness ratio: a) NSVIM b) BVNE

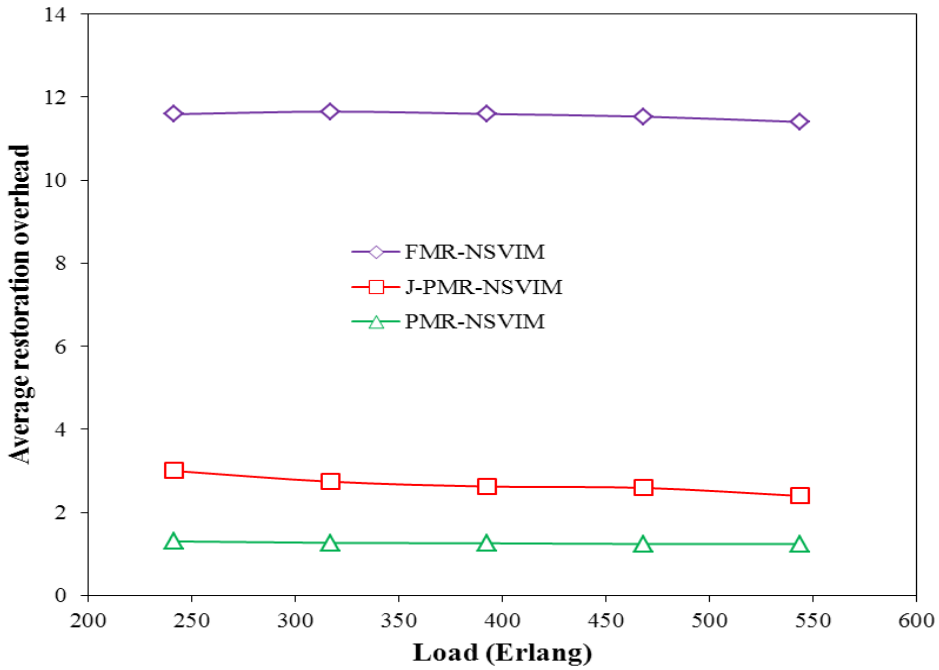



(a)

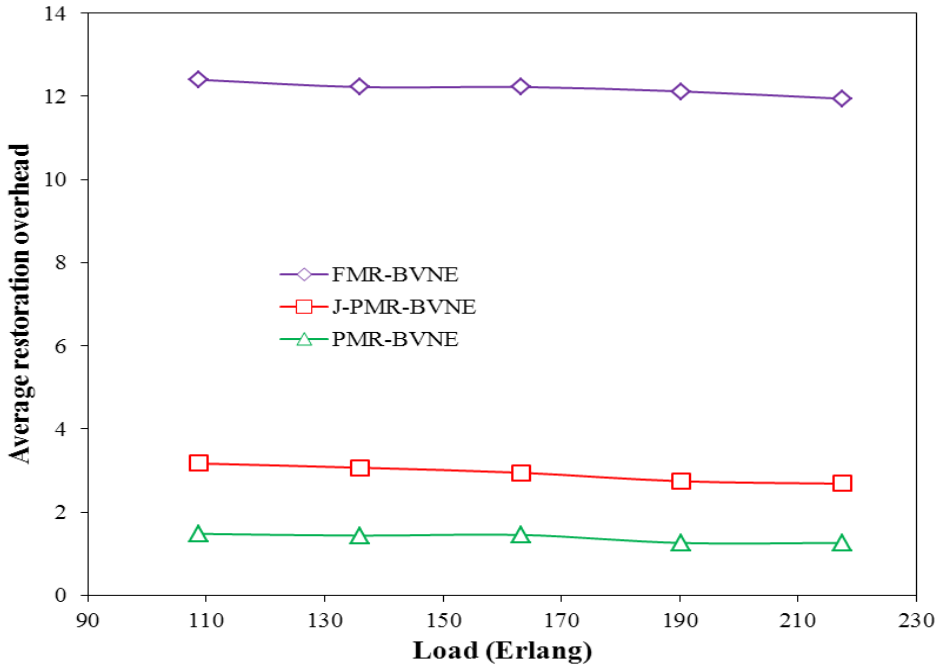



(b)

Figure 5.7: Fully-restored VN ratio: a) NSVIM b) BVNE




(a)




(b)

Figure 5.8: Restoration overhead: a) NSVIM b) BVNE

# Chapter 6

# Conclusions and Future Work

This dissertation presents a detailed study of survivability for cloud networking services. The work starts by presenting a background overview of some related work in Chapter 2. Subsequently, survivable VN embedding under multi-failure stressors is modeled as a MILP problem and two novel failure region-disjoint VN mapping heuristics are also proposed in Chapter 3. The more specialized case of VN embedding under probabilistic correlated failures is then treated in Chapter 4, and two heuristic schemes are proposed for minimizing the failure risk of a provisioned request. Finally, some alternate post-fault VN restoration strategies are developed to provide full and partial recovery from regional failure events in Chapter 5. The overall conclusions from these efforts are now presented along with some directions for future research.

## 6.1 Conclusions

This dissertation initially looks at survivable VN design for pre-defined static failure regions, i.e., stressors. Namely, a MILP formulation is outlined to pursue an optimal

strategy that minimizes the total cost of substrate resource usage. However, due to the complexity of this model, two further heuristic strategies are also proposed. In particular, the FRGBM scheme pre-partitions failure regions into two groups by placing geographically-closer failure regions into the same group. Regular VN mapping schemes are then applied to map VN requests onto the substrate so as to avoid each failure region group. Meanwhile, the DFRDM scheme dynamically computes the first VN mapping and then prunes its traversed failure regions before computing the second VN mapping. In addition, load balancing is also used to evenly distribute traffic loads across the substrate. Overall, some of the key contributions and findings from this study are as follows:

- The MILP solution is only solvable for relatively small network sizes, i.e., 10-12 nodes. However, this approach significantly outperforms all heuristic methods, particularly in terms of blocking rates and net revenues.
- The proposed FRGBM and DFRDM heuristics give very competitive results when compared with the only other known multi-failure protection schemes, i.e., the SOUM and IOCM strategies in [HY04]. In particular, these schemes yield significantly lower blocking rates and higher net revenues.
- Load balancing can significantly improve the performance of survivable VN mapping. This approach can be coupled with any existing VN mapping heuristic and almost always gives lower blocking and higher net revenues versus non-load balancing strategies (using fixed costs/weights).
- The DFRDM scheme gives slightly better performance than the FRGBM solution when load balancing is used, i.e., since this scheme dynamically computes two failure region-disjoint VN mappings.

In general, provisioning full protection for VN requests may lead to excessive resource usage, especially if catastrophic failure events are relatively rare. Hence

an alternate approach can look at modeling stressors as probabilistic events with specific substrate node/link failure probabilities and then leveraging this information to improve the resiliency of VN mappings. Along these lines, Chapter 4 presents two heuristic schemes to reduce the failure risk of VN requests. In particular, the RMM scheme only focuses on risk minimization, whereas the JRTM scheme jointly takes into account both risk minimization and TE efficiency concerns. Overall, simulation results indicate the following findings:

- The RMM scheme gives higher blocking rates than the base "non-risk" NSVIM embedding scheme. This is due to the fact that the former solution largely ignores resource efficiency concerns and chooses overly-lengthy routes for VN link connections. However, the JRTM scheme achieves a good tradeoff between these two alternatives, with blocking rates closer to the NSVIM scheme.

- Both the RMM and JRTM schemes yield much lower numbers of failed VN requests as compared to the "non-risk" NSVIM scheme. This is expected as the NSVIM scheme does not incorporate any probabilistic failure information into the VN mapping process. Moreover, the JRTM scheme closely tracks the RMM scheme in this metric as well, i.e., within 10%.

- The JRTM scheme gives the lowest VN failure rate among all the schemes, even below that of the pure risk-based RMM scheme. This improvement is due to the fact that this scheme has lower blocking rates for roughly equivalent numbers of VN failures, i.e., lower relative failure rates. Overall, these results confirm that jointly incorporating risk minimization and TE efficiency concerns can yield improvements over pure risk minimization strategies.

Finally, Chapter 5 looks at post-fault VN restoration for multi-failure stressors. Although these strategies cannot guarantee successful recovery in all cases, they are still very attractive for multi-failure scenarios. Along these lines, three different

restoration schemes are proposed here. Namely, the FMR scheme tries to fully recover all affected VN requests whereas the PMR and J-PMR schemes are more flexible and only pursue (partial) recovery of failed VN nodes and links. Overall, results show some key findings with regards to post-fault restoration:

- Post-fault VN restoration schemes provide significant improvement over pre-provisioned protection (FRGBM scheme). Namely, blocking rates are much lower and net revenues are also higher. This is expected as these solutions preclude any backup resource reservation.

- The FMR scheme achieves higher average VN completeness ratios than both the PMR and J-PMR strategies as it re-maps the whole affected VN request to new locations in the substrate. In general, this provides more recovery flexibility.

- The PMR and J-PMR schemes give lower restoration overheads than the FMR approach, i.e., measured in terms of VN node migrations. The reason here is that these strategies only recover the failed portion of a VN request and restore it in a best effort manner. The remaining unaffected parts (mappings) of a VN request are not altered.

- The J-PMR scheme outperforms the simpler PMR solution in terms of VN completeness ratios. This is expected since joint mapping of VN nodes and links is generally more resource efficient than separate two-stage mapping.

## 6.2 Future Work

This dissertation studies some important problems in the area of survivable cloud networking services. Overall, the findings and conclusions from this effort open up

promising new avenues for future research, some of which are now highlighted. First of all, one can consider further improvements to the heuristic strategies to try to close the performance gap with the MILP solution (Chapter 3). A key objective here can be to minimize backup resource usages. Some other possible strategies can also include relaxation techniques to solve the MILP for larger networks, improved resource sharing, and incremental backup provisioning.

Furthermore, as cloud services continue to expand, multi-domain VN provisioning and survivability concerns will also arise. Overall, this is an open problem area which needs further attention. Clearly, related designs will have to develop distributed/decentralized VN provisioning schemes, as it will not be possible to have complete "global" network visibility, i.e., due to carrier privacy and scalability concerns. Hence related efforts can study VN provisioning using abstract hierarchical routing state information.

Finally, the proposed VN survivability schemes can also be tested in real-world network testbed settings. The key goal here can be to identify any practical implementation bottlenecks and also validate performance in realistic environments. Of particular interest here would be to ascertain the best match between VN recovery methodologies and higher-layer cloud service types, e.g., IaaS, PaaS, SaaS, etc.

# References


[AB01] A. Belbekkouche, *et al.* "Resource Discovery and Allocation in Network Virtualization," *IEEE Communications Surveys & Tutorials*, Vol. 14, No. 4, 2012, pp. 1114-1128.

[AF01] A. Fischer, *et al*, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys & Tutorials*, published online: February, 2013.

[AG01] A. Greenberg, *et al*, "The Cost of a Cloud: Research Problems in Data Center Networks", *ACM SIGCOMM Computer Communications Review*, Vol. 39, No. 1, January 2009, pp. 68-73.

[BG01] B. Guo, *et al*, "A Novel Virtual Node Migration Approach to Survive a Substrate Link Failure," *National Fiber Optic Engineers Conference (NFOEC) 2012*, Los Angeles, CA, March 2012.

[BR01] B. Rimal, *et al*, "A Taxonomy and Survey of Cloud Computing Systems," *International Joint Conference on INC, IMS and IDC (NCM) 2009*, Korea, August 2009.

[CQ01] C. Qiao, *et al*, "A Novel Two-step Approach to Surviving Facility Failures," *OFC 2011*, Los Angeles, CA, March 2011.

[CZ01] C. Zhang, *et al*, "An Optimal Capacity Planning Algorithm for Provisioning Cluster-Based Failure-Resilient Composite Services," *IEEE International Conference on Services Computing (SCC) 2009*, India, September, 2009.

[DJ01] D. Justice, *et al*, "A Binary Linear Programming Formulation of the Graph Edit Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 8, August 2006, pp. 1200-1214.

[EO01] E. Oki, *et al*, "A Disjoint Path Selection Scheme with Shared Risk Link Groups in GMPLS Networks," *IEEE Communications Letters*, Vol. 6, No. 9, September 2002, pp. 406-408.

[FG01] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence," *Computers and Operations Research - Special issue: Applications of integer programming*, Vol. 13, No. 5, May 1986, pp. 563 573.

[GS01] G. Sun, *et al*, "Optimal Provisioning for Virtual Network Request in Cloud-based Data-centers," *Photonic Network Communications*, Vol. 24, No. 2, October 2012, pp. 118-131.

[GS02] G. Sun, *et al*, "Efficient Algorithms for Survivable Virtual Network Embedding," *Asia Communications and Photonics Conference and Exhibition (ACP) 2010*, China, December 2010.

[HK01] H. Kuhn, *et al*, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, Vol. 2, No. 1-2, March 1955, pp. 83-97.

[HL01] H. Lee, *et al*, "Diverse Routing in Networks With Probabilistic Failures," *IEEE Transactions on Networking*, Vol. 18, No. 6, December 2010, pp. 1895-1907.

[HY01] H. Yu, *et al*, "Virtual Infrastructure Design for Surviving Physical Link Failures," *The Computer Journal*, Vol. 55, No. 8, August 2012, pp. 965-978.

[HY02] H. Yu, *et al*, "Cost Efficient Design of Survivable Virtual Infrastructure to Recover from Facility Node Failures," *IEEE ICC 2011*, Japan, June 2011.

[HY03] H. Yu, *et al*, "Enhancing Virtual Infrastructure to Survive Facility Node Failures," *OFC 2011*, Los Angeles, CA, March 2011.

[HY04] H. Yu, *et al*, "Survivable Virtual Infrastructure Mapping in a Federated Computing and Networking System under Single Regional Failures," *IEEE-GLOBECOM 2010*, Miami, USA, December 2010.

[HY05] H. Yu, *et al*, "A Cost Efficient Design of Virtual Infrastructures with Joint Node and Link Mapping," *Journal of Network and Systems Management*, Vol. 20, No. 1, 2012, pp. 97-115.

[HY06] H. Yu, *et al*, "Migration Based Protection for Virtual Infrastructure Survivability for Link Failure," *OFC 2011*, Los Angeles, CA, March 2011.

[IF01] I. Foster, *et al*, "Cloud Computing and Grid Computing 360-Degree Compared," *Grid Computing Environments Workshop (GCE) 2008*, Austin, TX, November 2008.

[IF02] I. Fajjari, *et al*, "VNE-AC: Virtual Network Embedding Algorithm Based on Ant Colony Metaheuristic," *IEEE ICC 2011*, Japan, June 2011.

[JH01] J. Holland, Adaptation in Natural and Artificial Systems, *MIT Press*, 1992.

[JK01] J. Kennedy, *et al*, "Particle Swarm Optimization," *IEEE International Conference on Neural Networks 1995*, November 1995.

[JL01] J. Lischka, *et al*, "A Virtual Network Mapping Algorithm Based on Subgraph Isomorphism Detection," *ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures (VISA) 2009*, Spain, August 2009.

[MA01] M. Arregoces, et at, Data Center Fundamentals, *Cisco Press*, 2003

[MR01] M. Rahman, *et al*, "Survivable Virtual Network Embedding," *IFIPNETWORKING 2010*, India, May 2010.

[MY01] M. Yu, *et al*, "Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration," *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, April 2008, pp. 17-29.

[NC01] N. Chowdhury, *et al*, "Virtual Network Embedding with Coordinated Node and Link Mapping," *IEEE INFOCOM 2009*, Brazil, April 2009.

[OD01] O. Diaz, *et al*, "Network Survivability for Multiple Probabilistic Failures," *IEEE Communications Letters*, Vol. 16, No. 8, August 2012, pp. 1320-1323.

[RA01] R. Ahuja, *et al*, "Network Flows: Theory, Algorithms, and Applications," *Prentice Hall*, 1st Edition, February 1993.

[SK01] S. Kirkpatrick, *et al*, "Optimization by Simulated Annealing," *Science*, Vol. 220 No. 4598, 1983, pp. 671-680.

[SS01] S. Stefanako, "Reliable Routings in Networks with Generalized Link Failure Events," *IEEE Transactions on Networking*, Vol. 16, No. 6, December 2008, pp. 1331-1339.

[TG01] T. Guo, *et al*, "Shared Backup Network Provision for Virtual Network Embedding," *IEEE ICC 2011*, Japan, June 2011.

[TS01] T. Sttzle, *et al*, "MAX-MIN Ant System," *Future Generation Computer Systems*, Vol. 16, No. 9, June 2000, pp. 889 914.

[WP01] Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region, http://aws.amazon.com/message/65648/

[WY01] W. Yeow, *et al*, "Designing and Embedding Reliable Virtual Infrastructures," *ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures (VISA) 2010*, India, September, 2010.

[XC01] X. Cheng, *et al*, "Virtual Network Embedding Through Topology-Aware Node Ranking," *ACM SIGCOMM Computer Communication Review*, Vol. 41, No. 2, April 2011, pp. 38-47.

[XY01] X. Yu, *et al*, "Survivable Logical Topology Design for Distributed Computing in WDM Networks," *OFC 2009*, San Diego, CA, March 2009.

[XZ01] X. Zhang, *et al*, "An Overlay Mapping Model for Achieving Enhanced QoS and Resilience Performance," *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) 2011*, Hungary, October 2011.

[XZ02] X. Zhang, *et al*, "A Novel Heuristic for Overlay Mapping with Enhanced Resilience and QoS," *IET International Conference on Communication Technology and Application (ICCTA) 2011*, China, October, 2011.

[YC01] Y. Chen, *et al*, "Resilient Virtual Network Service Provision in Network Virtualization Environments," *IEEE International Conference on Parallel and Distributed Systems (ICPADS) 2010*, China, December 2010.

[YZ01] Y. Zhu, *et al*, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," *IEEE INFOCOM 2006*, Spain, April 2006

[ZZ01] Z. Zhang, *et al*, "A Unified Enhanced Particle Swarm Optimization-Based Virtual Network Embedding Algorithm," *International Journal of Communication Systems*, published online: January 2012.

[ZZ02] Z. Zhang, *et al*, "An Overview of Virtual Private Network (VPN): IP VPN and Optical VPN," *Photonic Network Communications*, Vol. 7, No. 3, May 2004, pp 213-225.

# Appendices

# Appendix A

# VN Mapping Heuristics

As noted in Chapter 3, the proposed FRGBM and DFRDM schemes can leverage any existing VN mapping heuristic. As a result, the recent NSVIM scheme is chosen and adopted here for the purposes of this thesis study, i.e., as it has been shown to outperform some other well-known solutions such as R-ViNE [NC01]. This algorithm is now detailed along with a further variation to perform substrate connectivity checks.

## A.1 Non-Survivable Virtual Infrastructure Mapping

The *non-survivable virtual infrastructure mapping* (NSVIM) scheme [HY04], [GS02] basically maps a VN request over a substrate network without regard to failures. This algorithm uses a single-stage approach to jointly map each VN node to a substrate node along with a subset of its attached VN links (substrate connection paths). This overall scheme is summarized in Figure A.1 and tries to minimize the total mapping

1: Set $MVN = \emptyset$, $ASN = \emptyset$, $UMVN = V_v$, $UASN = V_s$
2: Sort VN nodes in $UMVN$ according to their node degree
3: Choose $v_v$ with highest degree
4: Find candidate substrate node list $\mathbb{L}$ in $UASN$ according to node resource and bandwidth restrictions. If $\mathbb{L} == \emptyset$, return FAIL
5: Assign $v_v$ to $v_s$ in $\mathbb{L}$ with min. cost computed by Eq. A1-1
6: Reserve $r(v_v)$ node resource on substrate node $v_s$
7: For every VN link between $v_v$ and $v'_v \in M\text{-}adj(v_v)$, compute minimum-cost path $\mathbb{P}$, where each link in $\mathbb{P}$ has available bandwidth greater than or equal to the requested $b((v_v, v'_v))$. If no such path is found, return FAIL, else reserve bandwidth along $\mathbb{P}$
8: Move $v_v$ from $UMVN$ to $MVN$ and $v_s$ from $UASN$ to $ASN$
9: If $UMVN = \emptyset$, return SUCCESS. Otherwise, go to Step 2

Figure A.1: Non-survivable virtual infrastructure mapping (NSVIM) algorithm

cost. Consider the details here (using the same notation as in Section 3.1).

The NSVIM algorithm works by using two sets to track the mapped and unmapped VN nodes, i.e., $MVN$ and $UMVN$, respectively. Two additional sets are also defined to track the allocated and unallocated substrate nodes, i.e., $ASN$ and $UASN$, respectively. The scheme starts by initializing both $MVN$ and $ASN$ to empty and setting $UMVN = V_v$ and $UASN = V_s$. Now in order to map a VN node from the unmapped set, i.e., $v_v \in UMVN$, a candidate substrate node list, $\mathbb{L}$, is first built by selecting substrate nodes $v_s$ from $UASN$ based upon two criteria. First, the available nodal resources, $R(v_s)$, at a candidate node $v_s$ must be greater than or equal to the requested resources at the VN node, $r(v_v)$. Next, to ensure a feasible mapping, the maximum available bandwidth on all adjacent substrate links of $v_s$ must not be less than $b_{max}(v_v)$, i.e., where $b_{max}(v_v)$ is the maximum bandwidth requirement between

$v_v$ and the set of all VN nodes that are adjacent to $v_v$ in $G_v$. The latter set is denoted here as $adj(v_v)$, see Step 4, Figure A.1. Hence after $\mathbb{L}$ is constructed, the substrate node with minimum cost is selected as follows:

$$C(< v_v, v_s >) = C(v_s) + AC(v_s) + UAC(v_s) \qquad \text{(Eq. A1-1)}$$

where $C(v_s)$ is the cost of nodal resources used in substrate node $v_s$ and $AC(v_s)$ is the communication cost from $v_s$ to a subset of substrate nodes in $ASN$, i.e., which are allocated for VN nodes belonging to the set $M\text{-}adj(v_v) = adj(v_v) \cap MVN$. Detailed computations of the various terms in Eq. A1-1 are now presented as well.

First of all, the nodal resource cost for a VN node $v_s$, $C(v_s)$, is set to $\mathbb{C}(v_s)r(v_v)$ for a mapping $< v_v, v_s >$. Meanwhile, to compute $AC(v_s)$, the set of substrate nodes allocated to $M\text{-}adj(v_v)$ is defined in the set $SCMA$. Minimum-cost paths are then computed for each node pair $(v_s, v'_s \in SCMA)$ by using a shortest-path algorithm with link weights set to link costs, i.e., $C(e_s) = \mathbb{C}(e_s)b(e_v)$. The shortest path cost is then assigned as the sum of all link weights along the path. Hence, a total of $|M\text{-}adj(v_v)|$ paths are computed and $AC(v_s)$ is determined as the total sum of costs over all $|M\text{-}adj(v_v)|$ minimum-cost paths. Finally, $UAC(v_s)$ is defined as the estimated communication cost from $v_s$ to a subset of the substrate nodes in $UASN$, which may be allocated to VN nodes in the set $UM\text{-}adj(v_v) = adj(v_v) \cap UMVN$. In particular, this value is computed as follows. First, for every VN link $(v_v, v'_v \in UM\text{-}adj(v_v))$, a related minimum-cost path is computed from $v_s$ to as many nodes $v'_s \in UASN$ as possible. The average cost over these paths is then used as the estimated cost to map the VN link $(v_v, v'_v)$. If no such path can be found, the cost is set to infinity and $v_s$ is not deemed a feasible mapping. Using this, the $UAC(v_s)$ term is computed as the sum of the average costs for all VN links between $v_v$ and $v'_v \in UM\text{-}adj(v_v)$. Overall this term provides a "lookahead" capability and prevents premature mappings to low cost substrate nodes or links, i.e., since NSVIM is a greedy scheme and prior mapping choices may result in higher overall cost.

## A.2 Connectivity-Aware NSVIM

The *connectivity-aware NSVIM* (C-NSVIM) is a slightly-modified variant of above NSVIM scheme. Overall, the key difference here is an added constraint for computing the node list, $\mathbb{L}$, i.e., see Step 4, Figure A.1. Specifically, the maximum connectivity of a candidate node is also checked in addition to existing checks for node resources and adjacent link bandwidth constraints. This provision essentially tries to avoid mapping VN nodes to areas isolated by pruned failure regions. Hence if the maximum connectivity of a substrate node is less than the number of VN nodes in the VN request, this substrate node will not be considered as a valid candidate node to map this VN node $v_v$.

# Appendix B

# Risk Values for RMM Scheme

As per the failure model assumptions in Section 4.1, the risk of a substrate node (link) is related to the dot product of its risk vector $\vec{\boldsymbol{p}}_{v_s}$ ($\vec{\boldsymbol{p}}_{e_s}$) and the failure region probability vector $\vec{\boldsymbol{w}}$. Hence appropriate risk values are defined for nodes and links as $\xi_{v_s} = \vec{\boldsymbol{p}}_{v_s} \cdot \vec{\boldsymbol{w}}$ and $\xi_{e_s} = \vec{\boldsymbol{p}}_{e_s} \cdot \vec{\boldsymbol{w}}$. However, carefully note that the total risk of an end-to-end path is not linear with respect to the above risk values on its traversed nodes and links. As a result, regular shortest-path algorithms cannot be used to compute a minimum risk path by simply assigning node and link weights as $\xi_{v_s}$ and $\xi_{e_s}$, respectively. To address this concern, modified risk-based node and link weights need to be defined. A requisite solution is now presented along with a proof to show that these modified weights also minimize the path risk.

First, consider a survival rate definition for a substrate node and link, i.e., $\bar{\xi}_{v_s}$ and $\bar{\xi}_{e_s}$, respectively. Namely, these values represent the probability that the given node or link is not affected by a stressor event, i.e., $\bar{\xi}_{v_s} = \sum_{i=1}^{|U|} w_{u^i}(1 - p_{v_s}^{u^i})$ for node $v_s$ and $\bar{\xi}_{e_s} = \sum_{i=1}^{|U|} w_{u^i}(1 - p_{e_s}^{u^i})$ for link $e_s$. Now using these definitions, it can be formally shown that $\bar{\xi}_{v_s} = 1 - \xi_{v_s}$ as follows:

*Proof.*

$$\begin{aligned}
\bar{\xi}_{v_s} &= \sum_{i=1}^{|U|} w_{u^i}(1 - p_{v_s}^{u^i}) \\
&= \sum_{i=1}^{|U|} w_{u^i} - \sum_{i=1}^{|U|} w_{u^i} p_{v_s}^{u^i} \\
&= 1 - \sum_{i=1}^{|U|} w_{u^i} p_{v_s}^{u^i} \\
&= 1 - \vec{\boldsymbol{p}}_{v_s} \cdot \vec{\boldsymbol{w}} \\
&= 1 - \xi_{v_s}
\end{aligned}$$

□

Similarly, it can be shown that $\bar{\xi}_{e_s} = 1 - \xi_{e_s}$. Now using the above notation, the failure risk for a path $\mathbb{P}$ can be re-written as $1 - \prod_{v_s \in \mathbb{P}} \bar{\xi}_{v_s} \prod_{e_s \in \mathbb{P}} \bar{\xi}_{e_s}$. Hence, the objective for risk minimization along a path connecting two substrate nodes, $s$ and $d$, is given as:

$$\min_{\mathbb{P} \in S_{\mathbb{P}}} 1 - \prod_{v_s \in \mathbb{P}} \bar{\xi}_{v_s} \prod_{e_s \in \mathbb{P}} \bar{\xi}_{e_s} \qquad \text{(Eq. B-1)}$$

where $S_{\mathbb{P}}$ is the set of paths that connects substrate node $s$ and $d$.

Now further consider modified node and link risk values, i.e., $r_{v_s}$ and $r_{e_s}$, as defined in Eq. 4-3 and Eq. 4-4, re-presented here as:

$$r_{v_s} = log \frac{1}{1 - \xi_{v_s}} \qquad \text{(Eq. B-2)}$$

$$e_{v_s} = log \frac{1}{1 - \xi_{e_s}} \qquad \text{(Eq. B-3)}$$

Based upon the above definitions, it can be shown that the objective function in Eq. B-1 can be re-written as follows.

$$\min_{\mathbb{P}\in S_{\mathbb{P}}} \sum_{v_s\in\mathbb{P}} r_{v_s} + \sum_{e_s\in\mathbb{P}} r_{e_s} \qquad \text{(Eq. B-4)}$$

This is formally proved next.

*Proof.*

$$\begin{aligned}
\min 1 - \prod_{v_s\in\mathbb{P}} \bar{\xi}_{v_s} \prod_{e_s\in\mathbb{P}} \bar{\xi}_{e_s} &\iff \max \prod_{v_s\in\mathbb{P}} \bar{\xi}_{v_s} \prod_{e_s\in\mathbb{P}} \bar{\xi}_{e_s} \\
&\iff \max \sum_{v_s\in\mathbb{P}} log\bar{\xi}_{v_s} + \sum_{e_s\in\mathbb{P}} log\bar{\xi}_{e_s} \\
&\iff \min - \sum_{v_s\in\mathbb{P}} log\bar{\xi}_{v_s} - \sum_{e_s\in\mathbb{P}} log\bar{\xi}_{e_s} \\
&\iff \min \sum_{v_s\in\mathbb{P}} log\bar{\xi}^{(}_{v_s} - 1) + \sum_{e_s\in\mathbb{P}} log\bar{\xi}^{(}_{e_s} - 1) \\
&\iff \min \sum_{v_s\in\mathbb{P}} log\frac{1}{\bar{\xi}_{v_s}} + \sum_{e_s\in\mathbb{P}} log\frac{1}{\bar{\xi}_{e_s}} \\
&\iff \min \sum_{v_s\in\mathbb{P}} log\frac{1}{1-\xi_{v_s}} + \sum_{e_s\in\mathbb{P}} log\frac{1}{1-\xi_{e_s}} \\
&\iff \min \sum_{v_s\in\mathbb{P}} r_{v_s} + \sum_{e_s\in\mathbb{P}} r_{e_s}
\end{aligned}$$

□

Hence based upon Eq. B-4, the risk of path $\mathbb{P}$ can be represented as the *sum* of the modified risk values of each substrate node and link along this path. Hence, regular shortest-path algorithms can be used to compute the minimum risk path for VN links. Carefully note that since $\xi_{v_s} \in [0, 1]$, the range of $r_{v_s}$ and $r_{e_s}$ is also $[0, +\infty]$. Finally, it also can be shown that a substrate node with the minimum modified risk value, $r_{v_s}$, is equivalent to having the minimum original risk value $\xi_{v_s}$, as follows.

*Proof.*

$$\begin{aligned}
\min r_{v_s} &\iff \min log\frac{1}{1-\xi_{v_s}} \\
&\iff \min \frac{1}{1-\xi_{v_s}} \\
&\iff \max 1-\xi_{v_s} \\
&\iff \min \xi_{v_s}
\end{aligned}$$

□

Hence, this modified risk value, $r_{v_s}$, can also be used to minimize the risk for VN node mapping.